Data mining and statistical modeling for flight test applications in e-VTOL aircraft

Ali Tarabay

Supervised by Professor Michael Kokkolaras

Department of Mechanical Engineering

McGill University, Montreal

April 2024

A thesis submitted to McGill University in partial fulfilment of the requirements of the degree of Master of Science.

Abstract

Aircraft flight testing is an integral part of aircraft design and development. Flight testing serves as a method to expand and verify the aircraft flight envelope, as well as demonstrate adequate performance, stability, and systems compliance. It is a requirement of aircraft certification authorities to demonstrate aircraft safety and operability through the means of flight test.

In the past two decades, data collected during flight test has more than tripled as aircraft systems became more complex and sensor technology improved. In addition to the increased data collected per flight test hour, an aircraft program can require thousands of flight test hours before aircraft entry into service. Coupled with the associated high cost, the need for reliable classification and information mining from flight test data is apparent. This thesis will use data mining and statistical modelling techniques for the application of flight test data analysis in the context of designing an electric Vertical Takeoff or Landing (e-VTOL) aircraft. E-VTOL is a design concept that aims to improve the way that passengers and cargo traverse short distances. This concept leverages electric propulsion and combines the vertical takeoff and landing capabilities of rotorcraft with the forward flight efficiency of conventional aircraft. This thesis presents methods to solve part of the challenges that arise from flight testing this novel aircraft design.

Sommaire

Les essais en vol d'aéronefs constituent une étape essentielle dans la conception et le développement des aéronefs. Ces essais visent à étendre et à vérifier l'enveloppe de vol de l'aéronef, démontrant ainsi ses performances, sa stabilité, et sa conformité aux systèmes. Les autorités de certification des aéronefs exigent ces essais pour garantir la sécurité et l'opérabilité des appareils.

La quantité de données collectées lors des essais en vol a plus que triplé au cours des deux dernières décennies, en raison de la complexification des systèmes d'aéronefs et des améliorations technologiques des capteurs. En parallèle à l'augmentation des données collectées par heure d'essai en vol, un programme d'aéronef peut nécessiter des milliers d'heures d'essai avant sa mise en service. En raison des coûts élevés, la nécessité d'une classification fiable et d'une extraction d'informations à partir des données d'essais en vol est manifeste. Cette thèse utilise des techniques d'extraction de données et de modélisation statistique pour analyser les données d'essais en vol dans le cadre de la conception d'un aéronef à décollage et atterrissage vertical électrique (e-VTOL). Le concept e-VTOL vise à améliorer les déplacements de passagers et de marchandises sur de courtes distances. Exploitant la propulsion électrique, il combine les capacités de décollage et d'atterrissage vertical des hélicoptères avec l'efficacité du vol horizontal des avions conventionnels. Cette thèse présente des méthodes pour relever certains défis liés aux données issues des essais en vol de cette conception d'aéronef novatrice.

Acknowledgements

Firstly, I would like to acknowledge my supervisor Professor Michael Kokkolaras. His experience and guidance was instrumental in helping me navigate my research and the literature in the field. I firmly believe that I would not be able to submit my thesis if it was not for his trust and supervision style. I would also like to thank Professor Audrey Sedal for her thorough review and examination of my thesis and taking the time to provide extremely helpful feedback.

I would also like to acknowledge my colleagues who provided advice and challenged my ideas throughout this project. Namely Khalil Alhandawi from my research group who was always generous with his knowledge and experience in the field of optimization and machine learning. Josh Auberbach and Polly Mangan from Beta Technologies for their support and guidance, and for facilitating the use of my work at Beta for this research. I would also like to thank my colleagues at the Aircraft Performance department of Bombardier's Product Development division for their support and flexibility.

Finally, I would like to thank my family and friends for their love and support throughout my studies. They were always there when I needed them and supported me throughout this journey.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Aircraft flight testing is the primary method for design validation and eventual certification during aircraft development (1). A modern flight test vehicle can be equipped with hundreds of thousands of sensors collecting data at an increasingly large sample rate. In addition to the increased data collected per flight test hour, an aircraft program can require up to 5,000 flight test hours before the aircraft is in operation (2). This overwhelming amount of data necessitates the use of data mining and modelling techniques to help engineers efficiently make use of this data during the design and certification phases. Working with large quantities of data is increasingly common, and the challenges that arise from dealing with big data are well known and documented. However, what makes flight test data unique is the cost constraint. Not only is this data challenging due to its sheer volume, but also because of the prohibitive cost of collecting the data, where a single flight test hour can cost tens of thousands of dollars to conduct.

The overarching objective of this thesis is to propose and investigate methods for maximizing value extraction from flight test data. Various models, tools, and processes are

presented to facilitate data accessibility, analysis, and ultimately decision-making by the relevant stakeholders. The research will utilize flight test data and models from Alia, an electric Vertical Takeoff or Landing (e-VTOL) aircraft produced by Beta Technologies. This particular e-VTOL aircraft utilizes a design concept that aims to improve the way that passengers and cargo traverse short distances. By combining the vertical takeoff and landing capabilities of rotorcraft, and the forward flight efficiency of conventional aircraft, this concept can provide enhanced operation and mission capabilities relative to current aircraft on the market.

1.1 Aircraft Overview

It is worth spending some time describing the aircraft configuration as this aircraft will be the primary data source for the aircraft. Alia is an e-VTOL aircraft with a lift plus cruise configuration. This means that the aircraft is equipped with five electrically powered propellers. Four propellers are used for powered lift, these are used for aicraft maneuvering at low speeds (for example hover and vertical takeoff/landing). The fifth propeller is positioned at the tail of the aircraft in a pusher configuration and is used to propel the aircraft in forward flight at higher speeds. The aircraft has a 50ft wingspan and a range of 250 mi carrying 6 passengers. There also exists a version of this aircraft that is configured with only the electrically powered pusher propeller for Conventional Takeoff and Landing (CTOL). This version of Alia has a maximum range of 300nm (3). This aircraft provides an excellent data source as its configuration allows it to fly both conventional and VTOL mission profiles.

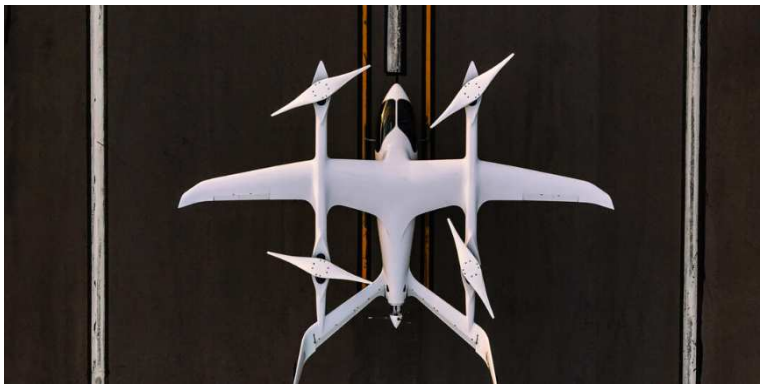

Fig. 1.1: Top view of the Alia Aircraft

1.2 Motivation

The definition of data mining used in this thesis is: *"a process for mapping low-level data (which are typically too voluminous to understand and digest easily) into other forms that might be more compact (for example, a short report), more abstract (for example, a descriptive approximation or model of the process that generated the data), or more useful (for example, a predictive model for estimating the value of future cases)*" (4). This definition introduced by Fayyad et al. in 1996 is still relevant today and adequately describes the objective of this thesis.

The main issue is that the magnitude and diversity of flight test data makes finding relevant segments of a flight a time-consuming task. Typically, engineers require specific maneuvers that satisfy certain flight conditions for their analysis. Currently the process for finding such maneuvers is rather manual. First, the engineer has to parse through the hundreds of flight test logs prepared by Flight Test Engineers (FTEs) after each flight. These logs are used to provide an overview of the different events that occurred during that particular flight as observed by the FTE. If a flight log indicates that this could be a relevant

flight, the aircraft configuration (weight, CG, installed systems, etc...) is checked to ensure that it meets the analysis requirements. Often, that aircraft configuration information must be retrieved from another database. Afterwards, the engineer parses the time history of the flight test and extracts the relevant portion of the flight that meets their criteria. This process is then repeated if the data is invalid, or if the analysis requires multiple demonstrations of a specific maneuver.

This process is not only iterative and manual, it can also hamper effective analysis. While flight testing is one of the best sources of information for understanding aircraft behavior, it suffers from all the issues inherent to real-world experimentation. Variation in operating conditions, pilot inputs, and aircraft condition can result in variance in the aircraft response. This variability can be assessed if access to large scale data is available. If the process of finding this data is sufficiently prohibitive, the engineer might be required to limit their analysis to one or two maneuvers that are 'good-enough', which might not accurately represent the underlying aircraft behavior. As will be seen in Section 4, a number of the models and workflows developed in this work can be used to identify and visualize long-term trends across various flights and flight phases.

As mentioned earlier, the explosion in the amount of data being recorded during flight test makes this manual process even more challenging. For reference, at Beta Technologies there has been roughly 900 billion data points recorded from the primary onboard data computer over the past year alone. There is an additional two data logging computers on the aircraft with a comparable quantity of data being recorded. The different logging computers provide data from differing sources covering a wide range of sensors, from strain gauge data to higher frequency vibration sensors. The data quantity and diversity issue is further exacerbated by the fact that sensor data is always recorded whenever the aircraft

is ON, regardless of any actual flight testing. Using the phase of flight model (discussed in detail in later chapters), we can estimate the amount of data where the aircraft is non-stationary to be 15% of the total data recorded. This 'needle in a haystack' situation is one of the main motivators for this research.

It is also clear that the challenges posed by big data are not unique to one particular organization or aircraft type. A single flight test on a Boeing 787 collects data from 200,000 multimodal sensors. The influx of sensor data along with ever-increasing code complexity on aircraft computers means that big data is becoming a real challenge (and opportunity) in the industry (5).

However, it should be noted that this abundance of data also presents an opportunity. While previously engineers had to rely on rule based logic to identify areas of interest (6; 7), this large quantity of data lends itself well to statistical methods and models that rely on large datasets for prediction and classification. At the same time, the problem of data quantity explosion coincides with unprecedented advances in computing, algorithmic statistics, and the wide-spread availability of open-source software providing data solutions (5). These conditions make a compelling case for investigating data mining and statistical modeling for flight test data application.

1.3 Thesis Objectives

As mentioned above, the main objective of this thesis is to present various approaches to maximize value extraction from flight test data. The focus is on how to quickly identify, extract, and present relevant portions of flight test data to engineers and stakeholders to allow for quick analysis and decision making. This is done via two main methods. The

first is the development of a statistical model to identify phases of flight during aircraft flight testing. As part of this exercise, various models, training and validation techniques, as well as performance metrics are investigated. In addition, an investigation into the use of a rule-based algorithm for maneuver identification is presented and some background on model hosting and data visualization is also provided. The second method builds upon the developed model, and uses other data processing techniques to define a process to allow analysts to easily join and search over flight test data from various sources.

As the model will be used to classify flight test data into separate flight phases, this use case will be used to benchmark the performance of different statistical models to determine the most performant model. This will provide an adequate application for building the infrastructure and data pipelines necessary for such an algorithm, as well as identify candidate models for more complex classification tasks in the future.

Finally, as described in the previous section, obtaining large amounts of pre-labelled flight test data (essential for most statistical and machine learning models) can be an expensive task. Therefore, the use of synthetic data obtained from high fidelity simulation models can provide an adequate substitute (8; 9). As a secondary objective, this project will provide an opportunity to assess the effectiveness of using synthetic data to fit statistical models for this application.

In summary, this research will aim to achieve the following objectives:

1. Assess the effectiveness of statistical models in classifying flight test data into discrete flight phases

2. Evaluate various candidate models and define metrics for performance assessment

3. Identify promising models for this task and potentially more complicated classification tasks in the future

4. Present hyperparameter optimization techniques suitable for different model types

5. Define a viable process for generating and using synthetic data for model fitting in flight test applications

6. Identify any special considerations that arise from using the aforementioned techniques with e-VTOL aircraft configurations.

7. Present a process to use model outputs to join on different data sources and make flight test data more accessible to end-users

Chapter 2

Theoretical Background

Most of the available literature that covers data mining and modelling techniques in aviation focus on operational safety, quality monitoring, and fault detection after aircraft entry into service (10). This is mainly due to the rigorous certification requirements that stress the requirement for repeatable, and explainable analysis during the aircraft design phase.

In operational safety applications, Random Forest models (such as the ones investigated in this thesis) are used to supplement Air Traffic Controller (ATC) priority assignment during aircraft taxiing conflicts (11). In this paper, the authors claim that the developed model can mimic ATC behavior with an accuracy of 89%. The paper also highlights the importance of feature engineering and provides a good example of how statistical models can be used in the aviation sector for operational purposes.

Another popular use of statistical modelling is in reliability and quality monitoring. Researchers from Airbus Canada and Polytechnique University in Montreal demonstrate how

the increased data output from modern aircraft can be used in reliability improvements of electric generators and fault scenario assessment for wiring harnesses. This paper highlights the use of Logical Analysis of Data (LAD), a supervised data mining technique. This is a pattern extraction technique that separates possible output scenarios into multi-classes representing different types of faults or failures given a series of input signals or physical attributes (12). One of the critical issues highlighted by this paper is the importance of transparency and interpretability in highly regulated and certified products such as aircraft. This is a main reason why the authors propose using LAD instead of other more common statistical modelling techniques such as deep neural networks. As discussed in Chapter 4, this requirement will also drive some of the model choices in this thesis.

Statistical models also feature in the literature for analysis during the design phase. As early as 1993, Linse and Stingel use a computational neural network to predict aircraft aerodynamics for Fly-by-Wire applications (8). More recently, with the resurgence of machine learning research, substantial research was conducted to investigate the use of machine learning techniques applied to flight data. To deal with the interpretability issues these models suffer from, Kirkpatrick et al. use a simple neural network with one layer and a linear activation function for system identification purposes. Furthermore, the advantage of using this simple neural network is that the model can be mapped to the A and B matrices of a state space representation of the system. Their results show that the model is able to quickly learn the underlying system of both longitudinal and lateral dynamics given a curated dataset of time histories of control input tests (13). In operation, Raman et al. from Honeywell Technology Solutions were able to train different ML models to predict tail specific performance degradation after aircraft entry into service. Factors such as aircraft age, route flown, and aircraft configuration were also taken into account to help with model

performance. One advantage of using aircraft data in operation, particularly for larger jets equipped with a sophisticated Flight Management System (such as the one developed by Honeywell), is the ability to extract phase of flight information from the flight plan input into the system. This allows data to be automatically segregated into separate phases of flight, assuming that the flight plan is adhered to (14). Wu et al. present one of the few papers involving flight test data, where the authors present a number of machine learning architectures that are meant to be used for anomaly detection in the flight testing of the COMAC C919 (15). However, the paper only proposes the architecture but does not go on to show the results of the modelling effort.

Even more relevant to this thesis, there are multiple examples in literature that present various frameworks and methodologies to classify flight phases. Tipps et al. from the University of Daytona worked closely with the Federal Aviation Adminstration (FAA) to analyse and process flight data into distinct flight phases (6). The analysis was done using commercial fleet data of the Boeing B-767-200ER and a similar analysis was done by the same team using Bombardier CRJ100 aircraft data (7). In this research, the authors relied on a rule-based logic to identify transition from one phase or another. This works well for classifying phases of flight, especially in commercial operation where the flight profiles are typically uniform with little variation. However, flight testing usually involves complex maneuvers and mission profiles that are highly irregular (as demonstrated in Chapter 3), making this approach ineffective for flight test applications. As before, neural networks feature prominently in the research for phase of flight classification as well. Researchers from the University of Hamburg worked with the German Aerospace Center to present a Long Short-Term Memory (LSTM) model trained on a mixture of publicly available data from Automatic Dependend Surveillance Broadcast (ADS-B) and simulation data to

predict aircraft phase of flight during flights in operation. The model demonstrated an improvement of over 2% compared to the state-of-the-art model (16). Kovarik et al. leveraged GPS and radar track data to train a phase of flight classifier for traffic conflict resolution and air space optimization. The data used for training contained 57 flights obtained from an Air Route Traffic Control Center. The paper shows that models such as Support Vector Machines (SVM) and LSTMs can be trained on this data for phase of flight classification of flights in operation (17). Both of these models are also investigated in the current work for flight testing phase of flight classification.

Research into rotorcraft flight data is also highly relevant. This is because the e-VTOL aircraft configuration in question is considered a hybrid between conventional and rotorcraft configuration. Rotorcraft also suffers from a similar heterogeneity of operation as rotorcraft mission profiles are relatively less defined. Chin et al. presents a number of different models that can be used for flight phase identification for rotorcraft data in operation (18). In this paper, eight different models are presented and compared for the classification task. The authors attempt to solve the issue of irregular mission profiles by splitting the problem into two sub-tasks. Flights are split into high-altitude and low-altitude regions where the possible flight phases are split between those two regions. This along with techniques such as sliding windows helps improve model performance. However due to the relative simplicity of the models used, only two out of the eight models investigated achieve a classification accuracy above 85%.

General aviation is another sector that features fairly irregular operation. In this domain, Li et al. take advantage of synthetic data by using a full flight simulator to generate training data for their model used to predict aircraft aerodynamic coefficients (9). Their model

shows promising performance when benchmarked against unseen synthetic data. However, model performance is not validated against any real flight data.

In summary, the literature in this area can be categorized into two categories. The first category takes advantage of large scale publicly available data to build statistical models for aircraft in operation. The second category is research in aircraft design and development that uses primarily synthetic data or curated and manually collected subsets of flight test data. This work aims to position itself between the intersection of data mining research, and research involving flight test data in aircraft design and certification. This is done by presenting a framework for flight test data classification and data mining. This framework is tailored to address the unique requirements of e-VTOL aircraft configurations.

Chapter 3

Considered Machine Learning Models

The following sections will focus on identifying a number of statistical models that can be used to classify portions of an individual flight test into discrete flight phases. In particular, the model will be designed to classify each data point in a flight test run into one the following phases:

1. Standing
2. Taxi
3. Takeoff
4. Climb
5. Cruise
6. Descent
7. Landing
8. Hover
9. Transition-In
10. Transition-Out

3.1 Datasets and Data Processing

As mentioned in Section 2, the literature contains a large amount of research focusing on classifying flight data in commercial operation. However, when reviewing the different flights in the available dataset, it is clear why the same approaches would not be applicable in a flight testing context. For example, Figure 3.1 below shows a relatively simple flight, with altitude on the y-axis and time on the x-axis. This particular profile was generated using the simulation model, and is comparable to the mission profile of a typical commercial flight.

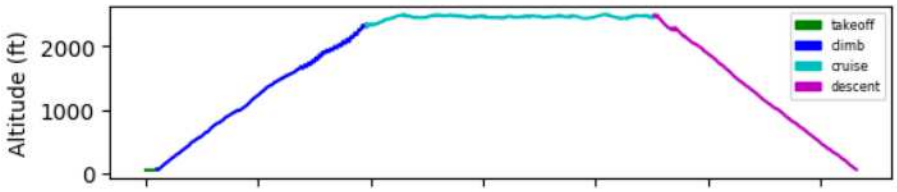

Fig. 3.1: Relatively Simple Flight Profile

However, the majority of the flights in the dataset are much more dynamic. As the primary objective of flight testing is to better understand and demonstrate the limits of the aircraft, a more complex mission profile is required. Figure 3.2 shows two mission profiles that are more representative of a typical flight test. In addition to the complexity of the mission profile, there is also a difference in data quality. This can be seen in the early portions of the second mission profile, where there is a clear discontinuity in the altitude data. This is because as the aircraft is being developed and tested, the associated sensors are also evolving. This means that sensor quality and calibration can vary greatly between flights and may provide lower quality data when compared to sensors in commercial operation.

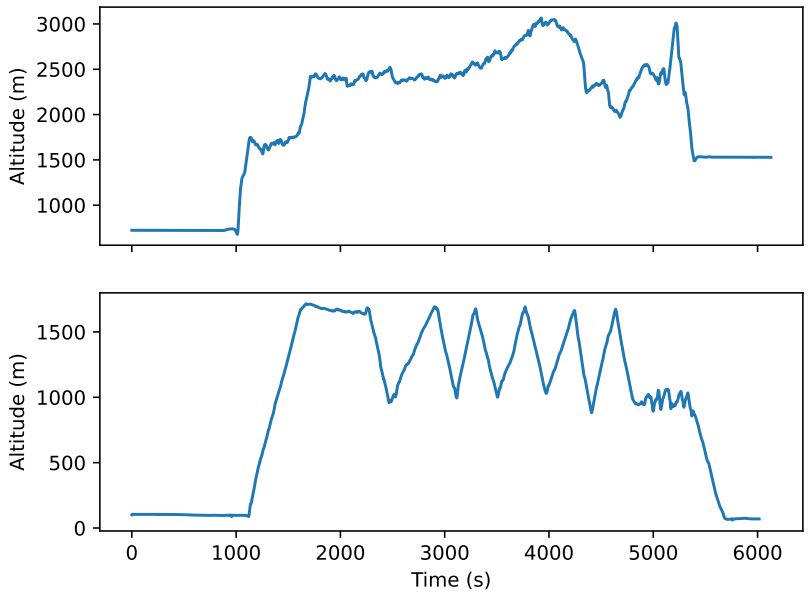

Fig. 3.2: Example of typical flight test mission profiles

Prior to model fitting, all data was down-sampled to 1Hz to allow for efficient data processing. This means that phase predictions are only accurate to the nearest second, which is appropriate for this use case. For phase of flight classification, it was initially determined that using four signals as model features is sufficient. The aircraft altitude, airspeed, heading, and pilot pusher throttle input were used to predict the aircraft phase of flight at each data point. The heading signal is particularly useful to help the model differentiate between the various on-ground phases. This is because before a takeoff or after a landing, the pilot will transition from the taxiway onto the runway or vice versa. This is usually associated with a large change in heading and can signal the beginning or end of the taxi phase.

Unlike more complex maneuvers that can be hard to find in flight test data, almost all flight tests will contain a large subset of the aforementioned phases of flight. Therefore,

creating a dataset of sufficient size using only real flight test data is relatively straightforward. The effect of supplementing this dataset with the generated synthetic data on model performance is discussed in Section 4.3.

Overall, the main dataset used for model development contained over 200,000 samples spanning 43 flights, where each sample represents one input instance into the model. An instance in this context can be an individual second with the corresponding 4 features, or a rolling window of multiple seconds each with 4 or more features (derived from the aforementioned signals). The choice of inputs is determined by the complexity of the model being fitted, where more complex models require less feature engineering on the input data. This is discussed in further detail below.

20% of the dataset was used for model evaluation and the remaining 80% was used for model training. The training subset was further split, where 20% was used for model validation and hyperparameter tuning. After hyperparameter tuning (see Section 3.4), the best model was trained on the entire dataset using 5-fold cross validation to get the model's average cross-validated performance.

Two methods of data ingestion into the model are investigated, the first involves ingesting each individual data point sequentially. In this approach each data point represents one second of flight test data (with 4 signals corresponding to altitude, speed, heading, and throttle input). The model will then classify each data point into the appropriate phase. This is sufficient for models that have memory-like attributes and can independently learn the relationship between sequential instances.

For model types that have no contextual or memory-like attributes, data will be grouped into sub-segments (referred to as windows from now on) each a few seconds long. A

statistical summary of each window (minimum altitude, average speed, etc.) is then fed into the model. This is meant to capture the temporal features of the data in order to aid the model with making the correct prediction. A rolling window technique is used for this approach.

Depending on the feature, different summary statistics can be used to capture the temporal information captured within each window. A simple average of the airspeed and altitude is used for each window. The total change in heading (i.e. sum of delta heading each second) was used for the heading feature and the maximum value of the pusher throttle feature were used as summary statistics. In addition, the change in speed and altitude over the window length are also added as features to the model. Finally, we can further augment the model inputs by extracting these summary statistics over windows of different lengths and using all of them as model inputs. For example, we can calculate these statistics for four different time windows of different lengths and feed those as different features into the model.

The optimal window length was determined through experimentation for each model and ranges from 10 seconds to 90 seconds depending on the model. For this approach, the target label of each window is the phase of flight at the window midpoint.

3.2 Synthetic Data Generation

The primary concern when generating synthetic data is that the the resulting dataset is fairly representative of the variability and variety of real-world data. As simulation models are typically deterministic, a stochastic element must be introduced when generating the various maneuvers and mission profiles. This is achieved by identifying a set of variables

that can be modified for each generated mission segment (cruise speed, wind gusts, autopilot controller gains, etc.). For each of these variables an upper and lower bound is defined and Latin Hypercube Sampling is used to generate different combinations of the variables for each simulation run. This sampling method is shown to generate sample representative of the design space for statistical model fitting (19). The parameters that are varied to generate these different mission parameters are listed below:

1. Cruise Altitude
2. Cruise Speed
3. Cruise Distance
4. Wind Gust Speed
5. Climb Speed
6. Descent Speed
7. Auto-pilot Gains
8. Field Elevation

The auto-pilot gains referenced above are PID gains of the Total Energy Control System (TECS) Controller used to fly the simulated mission profiles. The TECS controller is a Multi Input/Multi Output (MIMO) controller that maintains a vertical trajectory by constantly adjusting aircraft inputs to maintain a specific total energy for the system (20).

Using this method, a total of 46 synthetic flights were generated with various mission profiles. Figure 3.3 below shows a sample of the generated mission profiles.

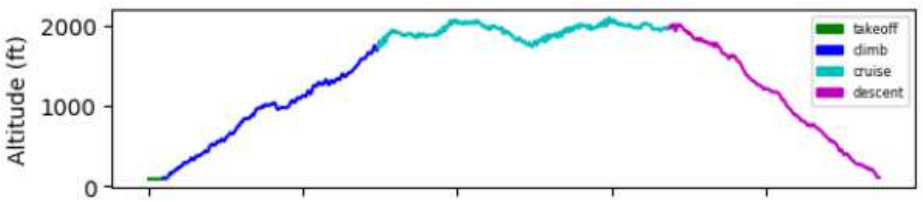

Fig. 3.3: Sample Simulated flight

The training algorithm was modified to only use the synthetic data for model training while model validation and testing was done using real flight test data to avoid skewing the results. However, due to simulation model and controller limitation, the mission profiles were modeled without the inclusion of a landing phase. This data was only used for the experiment described in Section 4.3.

3.3 Model Types

The different model types investigated for this project can be classified into 3 categories. The first category contains classic models that are applicable to both temporal and non-temporal data. As discussed in the previous section, these models require feature engineering to capture the temporal properties of the data to aid with classification. The 3 models investigated from this category are Support Vector Machinces (SVM), Random Forests, and Gradient Boosting models. These models were implemented using the Scikit Learn Python package (21). The package contains an implementation of a number of popular machine learning models along with utility functions that were used extensively in this research for data processing and pipelines, along with the Pandas Python Package (22).

The second category contains models that have properties that make them especially suitable for time-series data analysis. From this category, Long Short-Term Memory (LSTM), and Bi-directional LSTM models were investigated. These models were chosen because of their memory-like attributes that allow them to capture temporal dependencies and learn the context of each data point and its effect on the phase classification. The models in this category were implemented using the PyTorch Python package (23). This package provides a framework for deep learning model development, training and deployment. It

also provides native support for training on single or multiple Graphical Processing Units (GPU) which was used for this project to accelerate training time. The CUDA computing platform was used to take advantage of available GPU resources (24).

Finally, the third category involves models that do not have any memory-like attributes but require less feature engineering as they are able to learn the unique features associated with time series data. These models are able to identify and learn unique attributes of time series sub-segments without any manual feature engineering. These attributes can then be used to classify unseen data. From this category, the Shapelet Transform Classifier, and the Hierarchical Vote Collective of Transformation-based Ensembles (HIVE-COTE) model were investigated. These models were implemented using the Scikit Time Python package (25). This package contains implementation of machine learning models with a focus on time series problems. It is also compatible with the Scikit-Learn and Pandas Python packages that were used for data processing. While the intuition behind these models was very promising, their computational time complexity imposed a practical limit on their useability for this classification problem. Both algorithms had a time complexity of $O(n^2m^4)$ where n is the number of samples and m is the length of each window (26). This enforced a limit on the amount of data that can be used to fit the models as the algorithm run time scaled exponentially with these factors. For the Shapelet Transformer model only 10% of the available data was used, and for the HIVE-COTE model, only 1% (the equivalent of approximately 2000 samples) was used for model fitting. The effect of this limitation is reflected in model performance as discussed in Chapter 4. Each of the seven models is discussed in detail in the upcoming sections.

3.3.1 Support Vector Machine

Support Vector Machine (SVM) models use the idea of hyper-planes that can be constructed to separate different classes or types of values in a set in high-dimensional space. If the classes are linearly separable, this is a simple optimization problem that can be solved by maximizing the distance between the data point and a candidate hyperplane. This distance is referred to as the margin boundary. However, for non-linear problems (such as the current application), it is impossible to find a set of hyper-planes that perfectly divides the data into the correct classes. Therefore, a penalizing parameter is introduced to the optimization problem which incurrs a penalty whenever a data point is miss-classified or within the margin boundary (27). This is mathematically represented below:

$$\begin{aligned} &\min_{\omega,b,\zeta} \frac{1}{2}\omega^T\omega + C\sum_{i=1}^{n}\zeta_i \\ &\text{subject to } \; y_i(\omega^T\phi(x_i)+b) \geq 1-\zeta_i \\ &\zeta_i \geq 0, i = 1,\ldots,n \end{aligned} \tag{3.1}$$

x_i above represents a data point in the training set, and $y \in \{1,-1\}^n$. Where $y = 1$ represents a data point being correctly classified as part of its class and $y = -1$ represents a misclassification. ζ_i represents the distance between the data point and its correct margin boundary. C is a regularization parameter that acts as a psuedo-weight assigned to the penalty ζ_i.

For this model, the following parameters were used to obtain the results presented in this research:

Table 3.1: SVM Parameter values

Parameter	Value
Regularization Parameter 'C'	5
Kernel Coefficient 'γ'	0.25
Kernel	Sigmoid
Independent term	0.0
Tolerance	0.001
Class weight	balanced

3.3.2 Random Forest

Random Forest is an ensemble model that works by combining predictions from a number of decision trees to arrive at a class prediction consensus. In a Random Forest each tree is constructed using a bootstrap sample (random sample with replacement) from the training dataset. Each tree is further randomized by using a subset of available features to determine how nodes are split in a tree (28). This randomness results in reduced variance in the model prediction. In this implementation of the Random Forest model, the class prediction is determined by averaging the prediction of each classifier in the ensemble.

The following parameters were used to obtain the results presented in this research:

Table 3.2: Random Forest Parameter values

Parameter	Value
Number of Estimators	50
Maximum Tree Depth 'γ'	5
Quality Measure Criterion	Entropy
Minimum samples per leaf	1
Maximum leaf nodes	None
Class weight	Balanced
Minimum weight fraction per leaf	0.0
Minimum impurity decrease	0.0

As can be seen above, one of the practical limitations of the Random Forest algorithm is the large number of hyperparameters that can affect model performance. Section 3.4 summarizes the approach used to identify an optimal value for a subset of these hyperparameters.

3.3.3 Gradient Boosting

The Gradient Boosting model is another tree based ensemble model that is very similar to the Random Forest. However, unlike a Random Forest which independently constructs different decision trees, a Gradient Boosting model constructs the descision tree (or any other simple model) sequentially. The advantage of building the simpler models sequentially is that each model can be constructed to fit the gradient of the cost function being minimized. The result is that each subsequent simple model added to the base model

brings the optimization problem closer to the minimizer. Furthermore, model performance is improved by introducing an element of randomness, where in each iteration a subsample of the dataset is used to construct the simple model (29).

For this model, the following parameters were used to obtain the results presented in this research:

Table 3.3: Gradient Boosting Parameter values

Parameter	Value
Learning Rate 'α'	0.03
L2 Regularization	0.51
Leaf Node Minimum Samples	11
Maximum Tree Depth	7
Maximum iterations for boosting	100
Maximum bins	255
Maximum leaf nodes	31
Class weight	Balanced
Tolerance	1×10^{-7}

As in the Random Forest model, the large number of hyperparameters makes it difficult to identify an optimal set for the given classification problem.

3.3.4 Shapelet Transformers

Shapelet Transformer models classify time series data by identifying sub-sequences of the time series data to form discriminatory features. A shapelet represents a sub-sequence

of a time series data that is extracted by finding local phase independent similarity. The model works by calculating similarity measures for each candidate shapelet and then keeping the N shapelets with the highest information gain. The Shapelet Transformer then embeds the various shapelets into another model (such as a decision tree or an SVM) to form more complex rules. The first step involves defining the N shapelets that best represent the data. During classification, the algorithm calculates the similarity (using a distance measure) between the time series data and each shapelet. Combined with other models, this can be used to classify time-series data with high accuracy (26). However, as mentioned before, the algorithm time complexity prohibits the use of this model in practice.

The following parameters were used to obtain the results presented in this research:

Table 3.4: Shapelet Transformer Parameter values

Parameter	Value
Number of shapelet samples	1000
Maximum shapelets	400
Maximum shapelet length	None
Estimator	Rotation Forest
Batch size	100

Due to the high time complexity of this model, the number of shapelet samples and maximum shapelet parameters were chosen to minimize training time.

3.3.5 Hierarchical Vote Collective of Transformation-based Ensembles

Hierarchical Vote Collective of Transformation-based Ensembles (HIVE-COTE) is an ensemble model specifically developed for time series classification. It contains various classifier types such as bag-of-words dictionary based classifiers and even a shapelet transformer similar to the one presented in the previous section. The particular version of the HIVE-COTE model used in this research also includes a series of additional classifiers to further boost model performance. This model is known as HIVE-COTE 2.0 (30). In particular, this model contains a Driverse Representation Canonical Interval Forest (DrCIF) model. This is a tree based classifier that is particularly developed for identifying discriminatory features in time series model. (31). It also includes a Temporal Dictionary Ensemble model, a dictionary based model that transforms each time series into a historgram of word counts using a Symbolic Fourier Approximation (SFA) technique (32; 33). Finally the HIVE-COTE 2.0 model incorporates a Kernel based classifier known as Random Convolutional Kernal Transform (ROCKET). This classifier uses randomly parameterised convolution kernels to generate a feature vector for each time series (34).

HIVE-COTE model parameters were kept at their default values as it is an ensemble model of different classifiers. Only parameters that were modified were that of the Shapelet Transformer classifier, for that model parameters refer to Table 3.4.

3.4 Hyperparameter Tuning

Hyperparameter tuning is an essential part of machine learning model development. As model complexity increases, so does the effect of hyperparameters on model performance.

At first, tuning the various models was done manually, where for each model, different hyperparameter values were chosen based on intuition or on research of what has worked in the past. However, keeping track of the effect of each hyperparameter choice on performance for various model types simultaneously was becoming increasingly difficult. In addition to being inefficient and time consuming, there was always a concern of missing a more optimal combination of hyperparameters with this approach. Therefore, a more methodical approach was taken.

For the SVM, Random Forest, and Gradient Boosting models, a simple Grid Search algorithm was used for hyperparameter tuning. For each model, the most critical hyperparameters were identified and a Grid Search was performed to identify the optimal choice. For each possible combination of hyperparameter, the model performance was assessed based on recall performance (the significance of recall is discussed further in Section 4.1) and the most performant combination was used for final model fitting and evaluation. Table 3.5 below summarizes the different hyperparameters that were tuned for each of the three aforementioned models.

Table 3.5: Table of tuned hyperparameters for classic models

Model	Tuned Hyperparameters	Optimal Value
Support Vector Machines	Regularization Parameter 'C'	5
	Kernel Coefficient 'γ'	0.25
	Kernel	Sigmoid
Random Forest	Number of Estimators	50
	Maximum Tree Depth 'γ'	5
	Quality Measure Criterion	Entropy
Gradient Boosting	Learning Rate 'α'	0.03
	L2 Regularization	0.51
	Leaf Node Minimum Samples	11
	Maximum Tree Depth	7

While Grid Search Cross Validation was suitable for simpler models with shorter training time, it was impractical for the deep learning models due to the large computational cost during training. At the same time, these models were arguably more sensitive to hyperparameter choice. Therefore, two different optimization algorithms were used to help determine an optimal combination of hyperparameters. The average loss function evaluation on the validation subset of the data was used as an objective for the optimization algorithms to minimize.

These algorithms allow for a more efficient search and can reach a more optimal solution in less iterations. Two of these algorithms were tested for the LSTM and the bidirectional LSTM models. Using the Optuna Python package (35), Bayesian Optimization

based Tree-structures Parzen Estimator (36) and Non-dominated Sorting Genetic Algorithm (37) were adopted for hyperparameter tuning.

Tree-structures Parzen Estimator is an optimization algorithm that utilizes a probabilistic model to model the objective function. It uses this model to suggest the next set of hyperparameters. Starting with a few iterations of randomly selected hyperparameters, and evaluating them on the actual objective function, the estimator uses the results of these trials to improve on its probabilistic surrogate model. By splitting the results of its trials into a group that gave the best scores, and another that gave second best scores. kernel density estimators can be used to model the densities of both groups ($l(x_1)$ and $g(x_2)$ respectively). The next set of hyperparameters is sampled from the density model of $l(x_1)$, evaluating them in terms of $\frac{l(x_1)}{g(x_2)}$ and the set that returns the minimum value under $\frac{l(x_1)}{g(x_2)}$ is used to evaluate the original objective function (38).

The Non-dominated Sorting Genetic Algorithm II is a variation of Genetic Algorithms that provides a faster runtime and maintains elitism by using a paramater-less crowding approach, more details on this method can be found in Reference (37).

Both optimization algorithms resulted in very similar performance on the LSTM models (albeit with different hyperparameter combinations). In the end, using the Tree-Structures Parzen Estimator was favored due to its convergence proof, efficiency, and slightly better results. In addition to model specific hyperparameters, the optimization algorithm was also used to find the optimal window length to feed into the model. Table 3.7 below summarizes the different hyperparameters that were tuned for both LSTM models.

Table 3.6: Table of tuned hyperparameters for Deep Learning models

Model	Tuned Hyperparameters	Optimal Value
Bi-Directional LSTM	Learning Rate 'C'	0.00044
	Number of Layers	2
	Number of Cells	16
	Dropout	0.11
	Sequence Length	90 seconds
	Optimizer	Adam
LSTM	Learning Rate 'C'	8.2×10^{-5}
	Number of Layers	3
	Number of Cells	48
	Dropout	0.246
	Sequence Length	90 seconds
	Optimizer	Adam

Due to the high computational time complexity of the Shapelet Transformer and HIVE-COTE models, no hyperparameter tuning was performed on these models.

3.5 Model Parameters

In addition to the optimized hyperparameters obtained from the optimzation described in section 3.4, some parameters were used without further optimization. These parameters were found to have a smaller effect on model performance and so were excluded from

the optimization to limit problem size. Most of these parameters were set to their default values.

Table 3.7: Table of non-optimized hyperparameters for model definitions

Model	Hyperparameters	Value
SVC	Independent term	0.0
	Tolerance	0.001
	Class weight	balanced
	Maximum Iterations	None
Random Forest	Minimum samples per split	2
	Minimum samples per leaf	1
	Maximum leaf nodes	None
	Class weight	Balanced
	Minimum weight fraction per leaf	0.0
	Minimum impurity decrease	0.0
Gradient Boosting	Maximum iterations for boosting	100
	Maximum bins	255
	Maximum leaf nodes	31
	Class weight	Balanced
	Tolerance	1×10^{-7}
Shapelet Transformer	Number of shapelet samples	1000
	Maximum shapelets	400
	Maximum shapelet length	None
	Estimator	Rotation Forest
	Batch size	100

The HIVE-COTE model was not included in the table above as it is a collection of different classifiers (one of which is the Shapelet Transformer), each with its own parameters. With the exception of the Shapelet Transformer parameters described above, all other parameters for the three remaining classifiers were left at their default values described in (25).

3.6 Vertical Takeoff Or Landing Phase Identification

Similar to Conventional Takeoff Or Landing (CTOL) phases, having some means of Vertical Takeoff Or Landing (VTOL) phase identification can prove very useful for post-flight analysis. This is especially true because these phases are among the most critical phases, and drive a majority of design complexity (without these phases, the aircraft is a simple General Aviation aircraft with an electric powertrain). Moreover, in a flight test campaign spanning hundreds or thousands of hours, it can be difficult to find these phases that typically span a few minutes.

3.6.1 Phase Definition

For this work, the VTOL phases in question were aircraft hover, Transition-Out, and Transition-In. We can further define these phases as follows:

1. Hover: This phase begins when the aircraft's weight is no longer supported by the landing gear wheels. Lift is achieved solely using the aircraft lift motors. Hover phases ends at the application of power to the pusher propeller.

2. Transition-Out: This phase typically follows a hover phase and begins with the application of power to the pusher propeller. This phase ends when no power is provided to lift propellers and lift is provided solely by the aircraft wings.

3. Transition-In: This phase typically precedes a hover phase. It starts with the application of power to the lift propeller and ends when the aircraft ground speed is approximately zero. At this point, lift is primarily provided using the aircraft lift motors.

3.6.2 Phase Identification

As can be seen above, phase definitions are relatively well rigid. In addition, these phases are critical phases that push the aircraft design to its limit. Therefore it is unlikely that there will be a large dataset available for statistical model fitting or training. These two factors favor the use of a rule-based classification logic over classification using statistical model. Using the definitions above, a fairly simple but robust logic can be defined to classify these phases using only three aircraft signals. Namely Radar Altitude, Hover Throttle Input, and Pusher Throttle Input. Every data point where aircraft hover throttle input was above 50% and radar altitude was above three feet was classified as hover. In addition, to the above conditions, if the pusher throttle input is above 10%, the data point will be classified as a transition phase. Finally, additional logic is added to capture the ramp-up and ramp-down to the aforementioned values to ensure that the entire maneuver is captured. Figure 3.4 shows a decision tree representation of this logic. Note that the tree does not capture the portion of the logic designed to capture the ramp-up and ramp-down

of these phases. This logic was fairly effective in identifying VTOL phases despite some data quality issues (discussed in the 4.4).

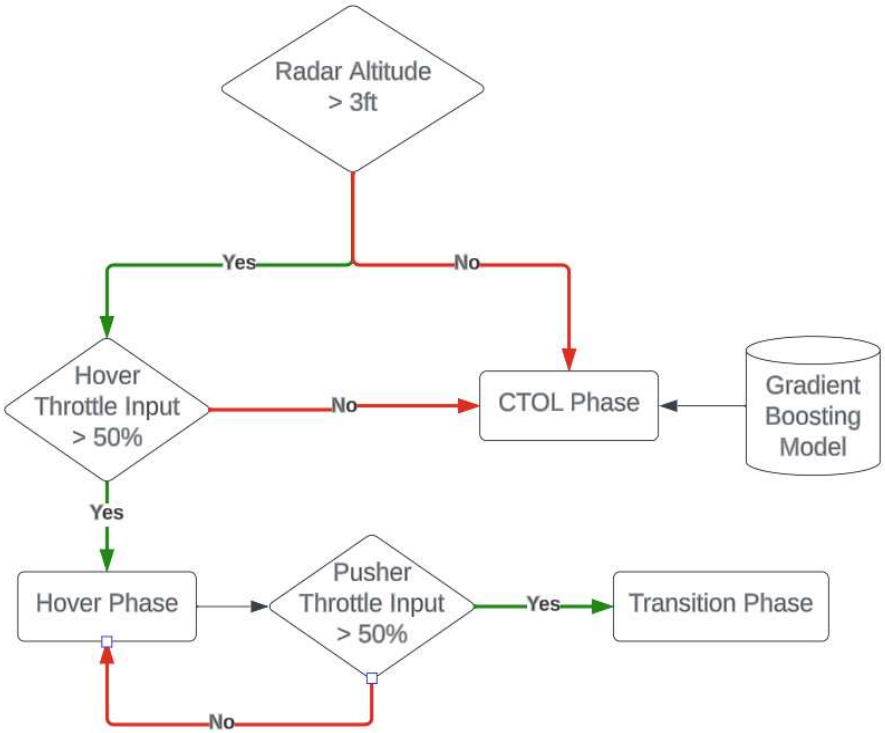

Fig. 3.4: VTOL Phase Identification Decision Tree

3.7 Model Deployment

A critical component of model success is the infrastructure used for model deployment and querying. If model classifications are not easily accessible, or if it takes too long to generate them, this can greatly reduce the model usefulness. To that end, the chosen model along with all the required pre-processing and post-processing code is wrapped in a Docker container and deployed on the Beta Technologies cloud platform. Containers are the industry standard for generating stand-alone code that runs quickly and reliably in different environments (39). This container contains the model, associated code, and all

required dependencies for running the data pipeline regardless of the runtime environment. This container is then hosted on SageMaker on Beta Technologies Amazon Web Services (AWS) account. In particular a Real Time Inference endpoint is used to host the container. This particular endpoint type allows the model to always be available and running waiting for an input query. The result is low latency and quick classification time, where over two hours of flight test data downsampled to 1 Hz can be classified in less than 60 seconds. The phase of flight classifications are then transformed by the post-processing code for ingestion into the same database where other aircraft information is also recorded. This makes joining telemetry data on phase classifications easy and relatively efficient. SageMaker also makes it fairly easy to implement status logging and alarm setting into the code for model performance tracking. This has proven to be quite useful when debugging issues, or to send alerts when an unexpected model prediction is made.

3.8 Flight Test Data Mining

One contextual layer above the classification into phases of flight for a given flight test, is a better understanding of the contextual information for an entire flight test. This includes vital information such as aircraft configuration and the conditions of the day. Information such as aircraft weight, CG, wind, and atmospheric conditions can have a substantial impact on aircraft behavior. Moreover, some analysis (for example stability and control analysis) is only applicable at specific aircraft configurations. This information along with other useful data such as flight test reports, operating pilot information, and FTE notes will be referred to as metadata in this work. Similar to determining the phase of flight, filtering and finding flight test data based on specific metadata can be very useful for analysts and engineers.

Moreover, since data is continuously being collected whenever the aircraft is turned on, simply determining when the aircraft is performing a flight test becomes necessary to filter out redundant data.

The main challenge with this metadata is that its not collected by aircraft systems. This is data that is typically documented by the FTE, sometimes days after the flight test is conducted. As a result, it is typically stored in human-readable forms in various sources in the organization. This then becomes an exercise in data wrangling where different metadata is collected from various sources and transformed to be stored in a location adjacent to the telemetry data collected from the aircraft. This process is outlined in the diagram below.

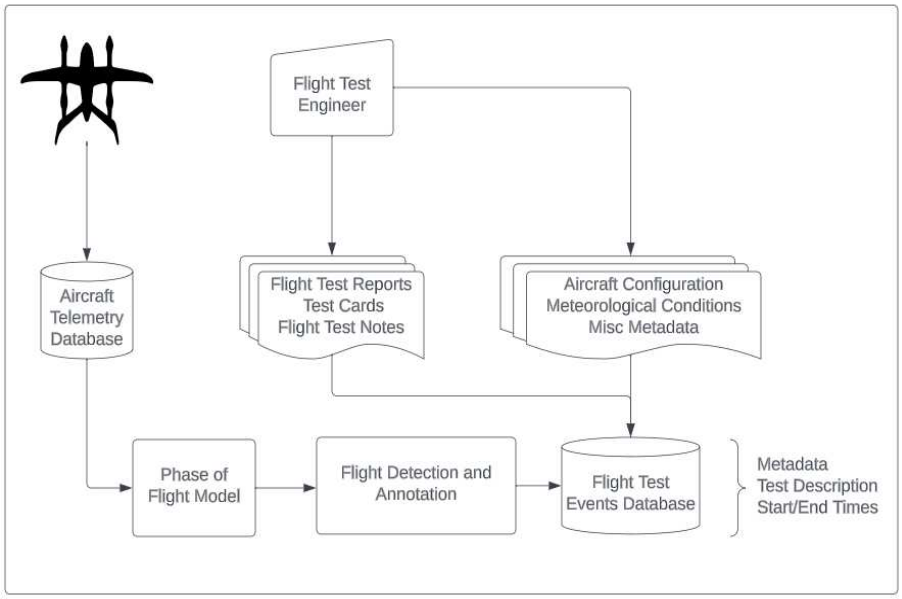

Fig. 3.5: Flight Test Data Mining Process

As data gets downloaded from the aircraft and uploaded to cloud storage, it gets ingested into the phase of flight model (described in previous chapters). Using the phase classifications from the model, the flight start and end times are identified. These times are identified as the first timestamp of the first takeoff of a flight (as classified by the model)

until the last timestamp of the last landing respectively. This indicates to downstream processes that there was a flight that occurred in this time range.

Artifacts such as flight test reports, test cards and flight test notes also usually have a timestamp associated with them (or at the very least a test date). We can match this temporal data to flight events identified by the phase of flight model. This connection means we can associate telemetry signals from the aircraft over a specific time range to particular flight test (and a flight test number) and its associated metadata.

A different data source is used for recording aircraft configuration (CG, weight, etc...) and meteorological conditions of the day. However unlike before, this data source uses a flight test number to identify this information instead of a timestamp. Therefore, we can use the information identified from flight test reports in the previous step to associate the flight test number with the flight time and connect the three different data sources together.

3.8.1 Data Sources

As shown in Figure 3.5, this process involves combining data from three different data sources to associate metadata with aircraft telemetry data. These different data sources are described in detail in this section.

Aircraft telemetry data is stored in what is called a data lake. A data lake is a data storage location where large amount of heterogeneous data from different sources is stored in its raw state. In general data in the data lake is schema-less but is partitioned in some way to aid in search-ability and to reduce query time (40). However, due to this schema-less heterogeneous nature, it is very difficult to achieve the query times that are sufficiently quick for dashboards and visualization purposes. Therefore, a subset of this data is transformed

into a structured SQL based database. Metadata is associated with aircraft telemetry data in both of the aforementioned datasets.

The second data source contains general flight test information such as reports, test cards, and FTE notes are stored in a simple cloud file storage system. The main challenge with this data is that its human generated and managed in a file system with almost no restriction on formatting and file structure. However, working with the flight test team, we were able to define an agreement for a consistent format for file structures. In particular, consistent naming conventions are used for associating files to particular tests using a flight test number identifier and a flight date. As long as the agreed upon naming convention is adhered to, aircraft telemetry data can be associated with the relevant flight test files.

The final data source is used to store key aircraft configuration data (weight, CG, etc. . .) along with the meteorological conditions of the day (dew point, wind, density altitude, . . .). This data is stored in Jira (41). In the Flight Test Jira project, each ticket represents a specific flight test. Tickets in this project contain the aforementioned aircraft configuration data and meteorological conditions for the relevant flight test. These three data sources are linked as described in Section 3.8.

3.8.2 Additional Considerations

Additional considerations have to be made to the above process to account for different practical issues that arise when this workflow was deployed. Firstly, there is the issue of incorrect model labels. Statistical models are not perfect, and since no model can be 100% accurate, some post-processing logic has to be implemented to ensure that the above workflow is not initiated due to an incorrect label. After model classification a number of

sanity checks are done to ensure that is not the case. First an aircraft signal is checked to make sure that the aircraft batteries are indeed providing high voltage power to the various systems. It is not possible that the aircraft will be in a non-standing phase if adequate power is not provided to the systems. Any non-standing phases identified by the model when power is not provided is not used to associate metadata. A second check is done to ensure that phase transitions is sensible. For example if the model predicts a cruise followed immediately by a takeoff, that is an indication that there was an issue with model prediction. In this case, an alert is generated and subsequent logic rejects the unrealistic phase.

The second issue arises from the sheer volume of data being collected at all times. There is over 8000 signals being collected from the aircraft at frequencies up to 1000Hz. This large amount of data requires the use of parallelization when ingesting and uploading this data. A side effect of not ingesting temporal data in sequence is that data can be ingested out of order. For example, the middle of the flight can be ingested first, followed by the end of the flight, and then finally the initial taxi and takeoff. This in turn means that the data is not ingested into the model sequentially. This is dealt with by modifying the logic to constantly scan ahead and behind the data currently being labeled by the model to expand the flight test event time bounds in case data gets ingested in an unexpected order.

Finally, as mentioned before, a lot of the data in this process relies on FTE input. However, due to human error and/or scheduling pressure, the data in these sources can be incorrect or incomplete. Therefore, in addition to connecting the different sources, the process above also scans for changes in the human input metadata ingested in the last 48 hours. If any of the metadata is modified in the original data sources, that change is propagated downstream.

Chapter 4

Numerical Studies

Model performance is typically reported in the literature in the form of model accuracy, usually reported against some benchmark dataset. While this provides an adequate measure of performance for generic tasks and applications, in practice it was found that high accuracy is not necessarily indicative of adequate model performance. The seven different models chosen for phase of flight classification were fitted and evaluated on the time series dataset using the split described in section 3.1. Table 4.1 and Figure 4.1 below shows a summary of the accuracy of the different models. However, despite the relatively high accuracy, most models were not suitable for this particular application. This was observed when reviewing model classifications across various flights in the test dataset.

Table 4.1: Summary of Model Accuracy

Model	Accuracy
Support Vector Machines	88.6%
Gradient Boosting	91.6%
Random Forest	84%
LSTM	92.9%
Bi-Directional LSTM	91.5%
Shapelet Transformer	86%
HIVE-COTE	49.3%

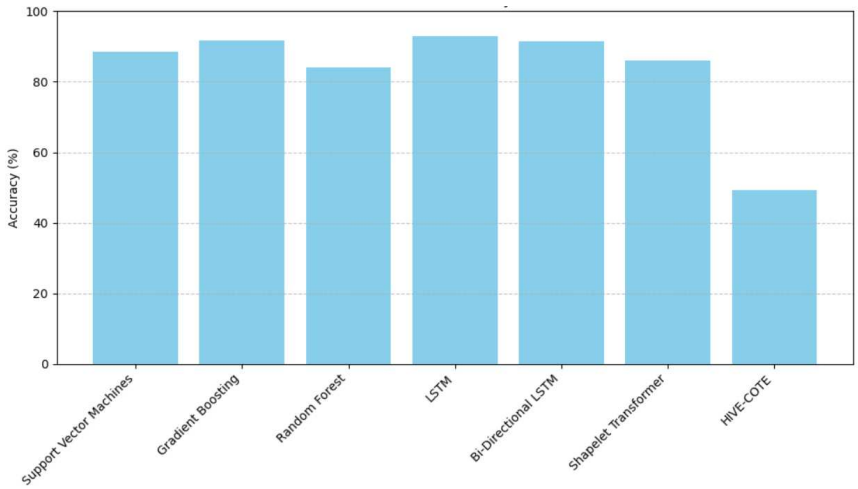

Fig. 4.1: Summary of Model Accuracy

Note: the accuracy above is calculated on the hold-out testing subset of the dataset. The final average cross-validated accuracy is reported in Section 4.2 for the subset of models that showed the most promise.

As can be seen in the table above, the HIVE-COTE model performance is degraded due to the dataset size constraint imposed by its high time complexity. We can also see

that almost all the other models achieve high accuracy of over 85%. However, as discussed in detail in the next section, it was found that accuracy is an not appropriate measure of performance for this classification task.

4.1 Evaluation Metrics

To demonstrate the ineffectiveness of accuracy as a measure of model performance for this classification task, Figure 4.2 shows the predicted label for each timestep using the Shapelet Transformer model on a sample flight from the dataset. The different colors in the plot correspond to different phase labels predicted by the model, shown against aircraft altitude on the y-axis and time on the x-axis.

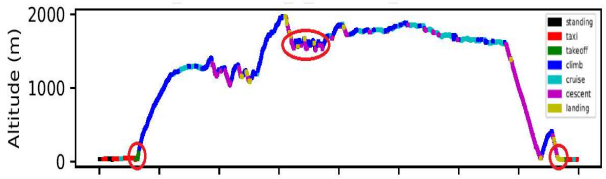

Fig. 4.2: Gradient Boosting Model Prediction - Sample Flight

Despite the high accuracy reported on the test dataset, reviewing the predictions in this plot shows a number of problems that hamper the effectiveness of this model. Firstly, the two red circles in the beginning and end of this flight highlight the takeoff and landing phases respectively (designated by green and yellow in the figure). These two phases represent the two most critical phases of flight. Therefore, it is essential that the model is able to accurately identify these particular phases with a large degree of accuracy. At the same time, these phases are very short relative to the length of the flight. In this particular flight, they represent approximately 1% of the total flight time. This means that a model can achieve a relatively high degree of accuracy (in theory up to 99%) and be quite ineffective

for practical purposes, as it fails to classify these critical phases. To mitigate this issue, takeoff and landing recall should be prioritized when comparing the performance of these different models.

Depending on the model in question, two approaches were used to maximize takeoff and landing recall for model predictions. For models that utilize a loss function as part of training, the loss function was modified by assigning weights associated with each class. A higher weight was assigned to takeoff and landing to ensure that the loss function evaluation further penalizes the missclassification of these phases during training.

For models where it was not possible to assign class weights to the loss function, this issue was addressed after model training. Since all the models used output a probability that a given timestep belongs to a specific class, it is possible to skew the model output to favor classification of takeoff or landing. This is done by enforcing these classes whenever the model predicts a probability for these classes that is above a specific threshold. This threshold was determined empirically to be 20%. This means that regardless of the other class probabilities, if the model predicts a 20% or greater probability that a given timestamp could belong to takeoff or landing, it is assigned one of those classes. If both probabilities are above 20%, the takeoff class takes precedence since it is deemed more critical. However, in practice, only in rare cases do takeoff and landings happen consecutively.

In addition, Figure 4.2 also highlights another issue towards the middle of the flight. The large fluctuations between phases during a relatively flat portion of the cruise could make the results of this model very difficult to work with. This means that when an analyst requests the phase classification for this flight, they will get a large number of segmented phases, each a few seconds long instead of a longer undisturbed cruise segment which is more useable for analysis.

In practice, it is highly unlikely that a cruise phase will last less than a minute. Therefore, after model prediction, it is possible to remove all cruise phases that are shorter than 60 seconds by merging them with the phases before and after. However, for phases other than cruise, it is possible to have shorter durations (for example, a takeoff takes approximately 45 seconds), therefore this minimum length threshold is reduced to 15 seconds for the other phases.

To account for these different considerations, and taking into account the initial accuracy values presented in Table 4.1, the following criteria was used to narrow down the number of models and help select the most performant model:

1. Takeoff and Landing Recall $> 70\%$
2. Overall Accuracy $> 90\%$
3. Predicted Phase Changes $<$ 1.25x Actual Phase Changes

4.2 Final Model Choice

Using the criteria defined in the previous section, 5 of the 7 models can be eliminated. This leaves only the Gradient Boosting model, and the bi-directional LSTM model. For those two models, hyperparameter tuning was performed along with a 5-fold cross-validation for performance evaluation. The 5-fold cross validation meant that each model was trained 5 different times and evaluated against the hold-out subset. All performance reported in this section is averaged across the 5 different folds. Although this meant that training now takes 5 times longer for each model, it is worthwhile since this reduces any

bias introduced by using a single subset for evaluation. The two models were tuned using the techniques outlined in Section 3.4

In addition to hyperparameter tuning, more feature engineering was conducted for the inputs into the Gradient Boosting Model to aid its performance. Similar to before, a window of time was used to extract summary temporal features. However, for this model these features were extracted over multiple time windows (10 seconds, 30 seconds, and 60 seconds) and concatenated for model input. This was not necessary for the LSTM model as it is already capable of learning the defining features of each phase over multiple time windows. Table 4.2 below summarizes the tuned Gradient Boosting model performance using the aforementioned metrics averaged over a 5-fold cross validation evaluation of the entire dataset.

Table 4.2: Gradient Boosting Model Performance

Overall Accuracy	94.8
Takeoff Recall	94.5%
Landing Recall	83.3%
$\frac{Predicted\ Phase\ Change}{Actual\ Phase\ Change}$	1.27

Table 4.3 summarizes the tuned bi-directional LSTM model performance results using the same process.

Table 4.3: Bi-Directional LSTM Model Performance

Overall Accuracy	94.5
Takeoff Recall	92.6%
Landing Recall	71.6%
$\frac{Predicted\ Phase\ Change}{Actual\ Phase\ Change}$	1.14

As can be seen above, both models achieved comparable results, with the Gradient Boosting model performing relatively better. Interestingly, both models achieved a relatively low landing recall score despite the special attention that was given to this phase. However, even compared to the takeoff phase, classifying a landing is particularly difficult. The landing phase is defined to begin at 50ft above the ground. However, until the aircraft landing gear touches the runway, that phase is almost indistinguishable from the descent phase. Both phases have a relatively similar change in altitude and a constant speed, there is also no change in throttle input as the motor is at idle or low power setting. This is in contrast with a takeoff phase which is distinguishable from the phases that typically come before and after. The takeoff can be easily distinguished from a taxi phase due to the sudden increase in speed and eventually altitude. It can also be distinguished from the climb phase that typically precedes it, as the aircraft in climb will reduce its acceleration and stabilizes at the required climb speed.

It could be argued that the LSTM model performance could be further improved by more feature engineering (similar to what was done for the Gradient Boosting model). However, the advantage of using more complex models such as an LSTM model for this simple task is that it should not require a large degree of feature engineering. This reduced pre-processing of the data comes at the cost of longer training time, less explainable results,

and a more complex model. Therefore, the Gradient Boosting model was found to be the most suitable model for this application. Figure 5 shows a sample prediction of the Gradient Boosting model. As can be seen in the figure, both the takeoff and landing phases are well captured. In addition, the model classification is not fluctuating between different phases resulting in a low number of phase changes relative to the actual number of phase changes in this flight. Even more impressive, is the model ability to distinguish between a go-around (an aborted landing where the aircraft never touches the runway), and a touch-and-go (a landing immediately followed by a takeoff).

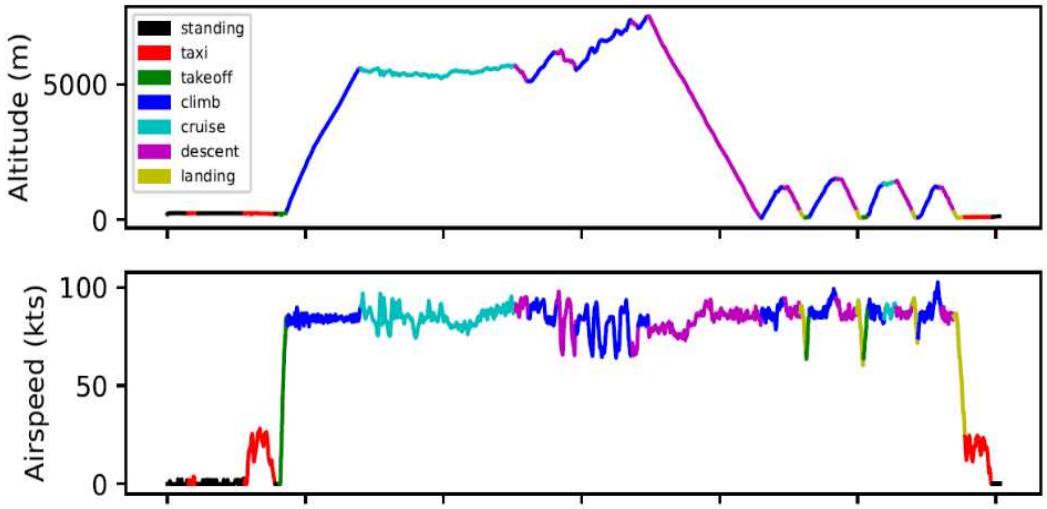

Fig. 4.3: Gradient Boosting Model Prediction - Sample Flight

4.3 Use of Synthetic Data for model training

Appending synthetic mission profiles to the training dataset was mainly done to investigate if the LSTM model performance can be improved by increasing the dataset size. Recurrent Neural Network models (such as LSTM) are notoriously data hungry due to their large number of parameters. Therefore, it was expected that increasing the size of

the dataset would have a positive impact on model performance. However surprisingly, it was found that model performance was degraded when appending the synthetic mission profiles to the training dataset. A summary of the performance degradation can be found in the table below.

Table 4.4: Performance deltas due to synthetic training data

Metric	Delta
Accuracy	-5.32 %
Takeoff Recall	-3.60 %
Landing Recall	-17 %
$\frac{PredictedPhaseChange}{ActualPhaseChange}$	+8.3%

All performance metrics with the exception of the phase change ratio deteriorated with the addition of synthetic mission profiles to the training dataset. This could be an indication that the synthetic mission profiles are not adequately representative of mission profiles seen in real flight test data. As mentioned in Section 3.1, flight test mission profiles need to be fairly complicated to exercise the aircraft design. This variation and complexity can be difficult to replicate using automatically generated synthetic data. Finally, the large deterioration in landing recall performance is likely due to the fact that the synthetic mission profiles did not include the landing phase. As a result, the amount of landing phases present in the dataset relative to other phases was reduced. This had the unexpected result of disincentivizing the model to 'learn' the identifying features of this phase due to its relatively small number.

4.4 VTOL maneuver identification

VTOL maneuvers were identified using a set of rules that codify the phase definitions identified in Section 3.6.1. These set of rules were found to correctly identify hover phases in 86.9% of the time. There was not enough transition data to present an accurate estimate of model accuracy.

One immediate shortcoming of relying on rule-based logic compared to more complex models is the effect of data quality on model performance. Discrete rules are not very robust to noise in the data or signal drops. However as described in Section 3.6.1, 2 out of 3 signals (pusher and hover throttle inputs) required for VTOL phase identification come directly from inside the aircraft. These signals are much less susceptible to data quality issues. However, Radar Altitude is much more susceptible to signal noise. Radar Altitude is determined by sending radio waves of differing frequency from the airplane and measuring frequency differences in their reflection off the ground to determine aircraft height (42). This makes the signal fairly susceptible to data quality issues. Possible workarounds for this issue are discussed in 5.1.

4.5 Application and Sample Use Cases

The model described in the previous sections is deployed and accessible for Beta Technologies engineers. Since its deployment, almost all flights conducted on the program's flight test vehicles have been ingested by the model for phase classification. The model consistently generates reliable phase labels that are used in downstream visualisation and analysis. This section presents some flights demonstrating model strengths and weaknesses,

as well as some sample analysis use cases demonstrating the usefulness of the model for different engineering disciplines. It should be noted that all aircraft data has been obfuscated while maintaining trends wherever possible. This is done to ensure that confidential aircraft performance data is protected while still demonstrating the added value of the work presented in this thesis.

The Figure below shows a sample flight from a flight test vehicle in August 2023. The first two panels in the plot show the aircraft altitude and speed respectively, the third panel shows the pilot throttle input throughout the flight. Towards the end of the flight, a mixture of low approach and touch-and-go maneuvers are conducted. As can be seen from the different plotted signals, it is difficult to visually differentiate these maneuvers. However, after discussing the flight profile with the FTE responsible for this particular flight test, model classifications were confirmed to be accurate. This indicates that the different strategies used to ensure these critical maneuvers are captured were effective.

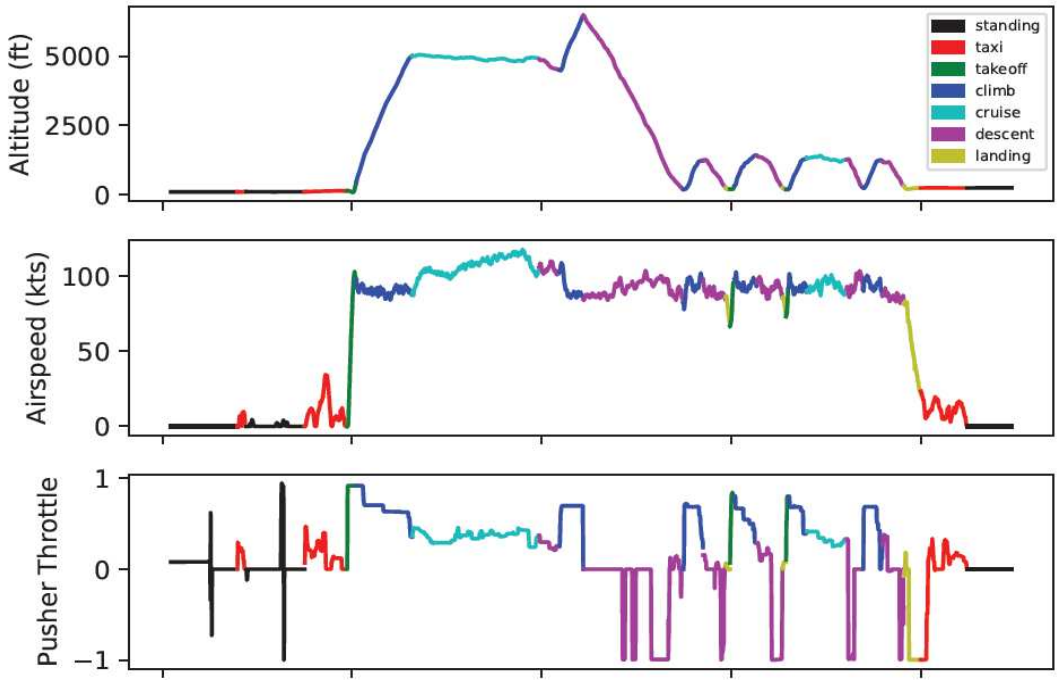

Fig. 4.4: Sample Flight Test - Low Approach vs. Touch-and-Go

On the other hand, one issue that was discovered shortly after model deployment, is the model's inability to differentiate cruise phases from climb and descent phases with relatively low rates of climb/descent. The flight test shown in Figure 4.5 shows an example of this. As seen below, the aircraft goes into a shallow descent immediately after the initial climb. However, due to the low rate of descent, the model incorrectly classifies this as a cruise phase even though there is a substantial decrease in altitude over the length of the phase. The incorrect classification is believed to be caused by two main issues in the training dataset. Firstly, only a few flights in the training dataset contained mission profiles where the aircraft performed shallow climbs and descents. Moreover, the phase definition used when labeling flights for the dataset dictated that any climb or descent with a rate less than 100fpm would be labeled as cruise. In the next iteration of model training, this threshold will be reduced to 50fpm. This change, along with ensuring that more tests with mission profiles flown in this manner are present in the training dataset is expected to mitigate this issue.

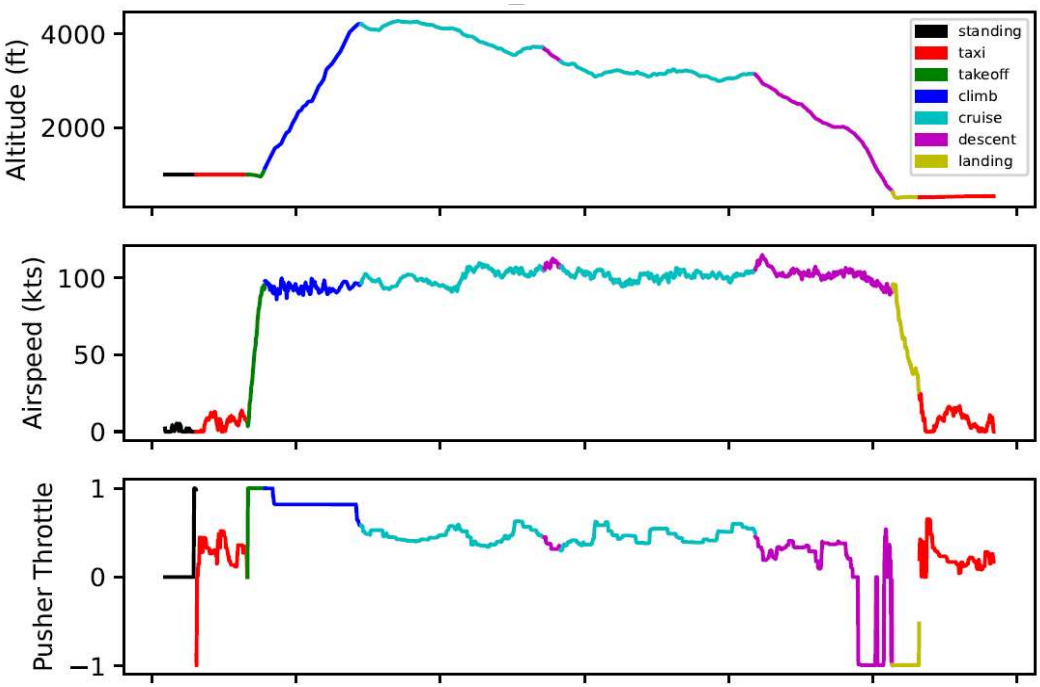

Fig. 4.5: Sample Flight Test - Low Rate of Descent

4.5.1 Sample Use Case - Pilot Induced Oscillation

As the name implies, Pilot Induced Oscillation (PIO) occurs when the aircraft enters an uncontrollable oscillatory state due to pilot inputs, usually in the opposite direction of the aircraft's current state in an attempt to control it. This dangerous handling issue has been the cause of multiple accidents over the years (43).

A recent update to the aircraft control systems triggered an analysis to better understand the aircraft's susceptibility to PIO during landing. The purpose of the analysis was to determine if special training procedures were required for pilots to ensure that the aircraft remains controllable during the landing phase. For that effect, it was necessary to study pilot inputs across multiple landings to account for variability due to weather, glide slopes and pilot in control. Getting this data manually would require substantial effort. First

the analyst would have to download the data for each flight test, plot the flight, and then manually identify the start and end time of the landing phase for each flight test. That portion of the test would then need to be extracted and then aggregated with all the other landings in order to create a comprehensive understanding of pilot behavior during the landing phase. This can all be achieved using a few lines of Python code and a Structured Query Language (SQL) query thanks to the phase of flight model developed as part of this research.

For this analysis, pilot inputs for 101 landings (as identified by the model) were collected. For each landing, the different pilot inputs were plotted against aircraft altitude and time to better understand pilot inputs across the various stages of the landing phase. Figure 4.6 shows a sample of these plots representing 3 different landings from the first half of 2023.

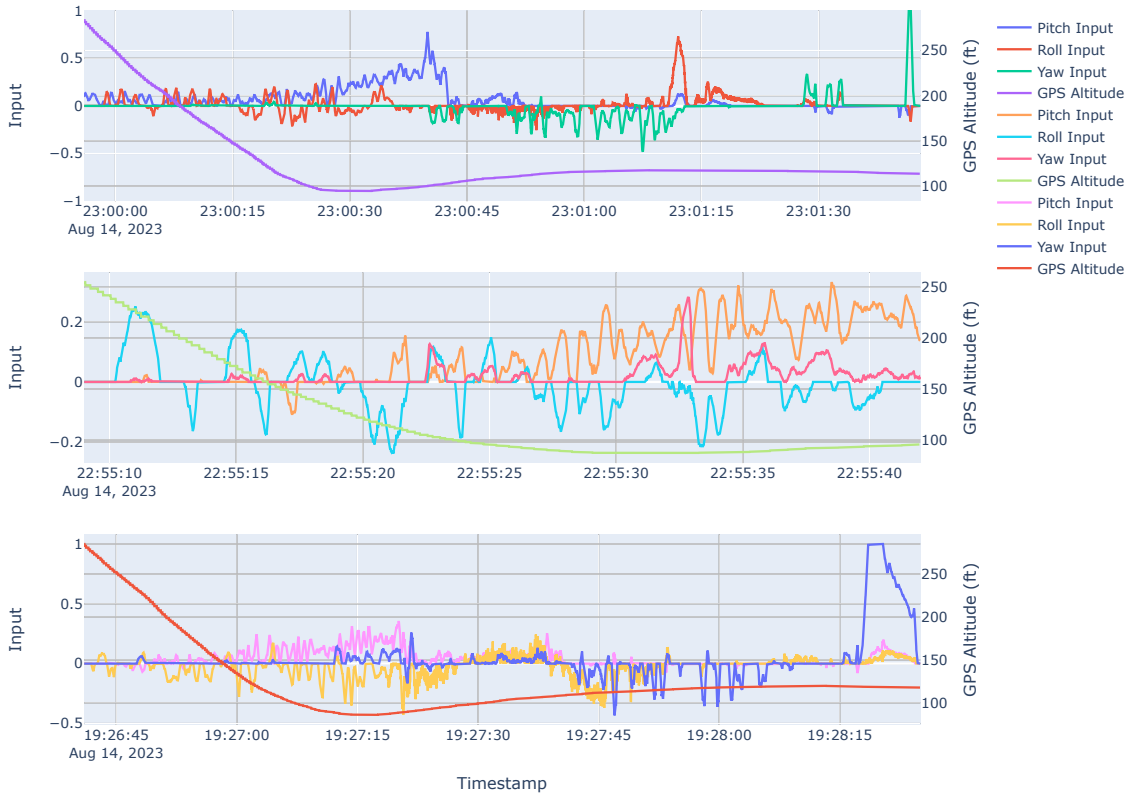

Fig. 4.6: 3 Sample Landing plots - PIO investigation

PIOs are typically induced when the pilot inputs alternate in the opposite direction to aircraft orientation in attempt to control the aircraft. However, as can be seen in the plots above there is very little oscillation in pilot inputs. The only exception is the yaw input but that oscillation seems to occur after aircraft touchdown (as indicated by the lack of pitch input).

4.5.2 Sample Use Case - Diagnostics and Trend Monitoring

Another useful capability that arises from automatic phase classification is the ability to conduct large-scale longitudinal analysis. The benefit of this is two-fold. Firstly, critical signals can be monitored and flagged if found to be outside the normal range for a given phase of flight. The second benefit is the ability to determine trends and better understand aircraft performance. This is done by plotting signals of interest for a given phase across a number of flights. Three use cases satisfying this purpose are presented below.

As mentioned before, takeoff is one of the most critical phases of flight. During this phase, aircraft components such as the motors and batteries undergo the most stress. Therefore, it is critical to monitor both motor and battery temperatures during this phase to ensure that they are within normal bounds. Since takeoffs are automatically classified, a single SQL query can be used to extract motor and battery temperatures for all the takeoffs over a specific time range. The maximum temperature and the rate of temperature change can be calculated for each takeoff and inspected to highlight any outliers or concerning trends. Table 4.5 shows a subset of the takeoff data collected from running this query over a two month period. As can be seen, there are no obvious outliers in the data and battery and temperatures are within the expected range.

Table 4.5: Motor and Battery Temperatures During Takeoff

Date	Motor Max Temperature (C)	Motor Temperature Rate of Change (C/s)	Battery Max Temperature (C)	Battery Temperature Rate of Change (C/s)
2023-08-06	418.74	0.11	157.64	0.02
2023-08-08	352.59	0.06	188.07	0.08
2023-08-09	389.42	0.03	212.52	0.01
2023-08-17	352.29	0.05	178.34	0.01
2023-08-28	520.98	0.06	157.08	0.02
2023-08-31	529.23	0.05	184.06	0.04
2023-09-01	493.53	0.06	178.56	0.02
2023-09-06	569.25	0.06	205.38	0.03
2023-09-07	544.20	0.07	201.22	0.01
2023-09-12	264.44	0.07	214.08	0.02
2023-09-21	505.41	0.07	174.73	0.15
2023-10-03	511.45	0.08	157.68	0.03

Figure 4.7 is another way to visualize the data. In this view, one interesting trend that is visible is the motor temperature increase during touch-and-go maneuvers. This is demonstrated by the vertically aligned dots in the plot corresponding to takeoffs from a flight on August 31st. During this flight, a number of touch-and-go maneuvers stress the aircraft motors and batteries significantly causing its temperature to rise with each successive takeoff.

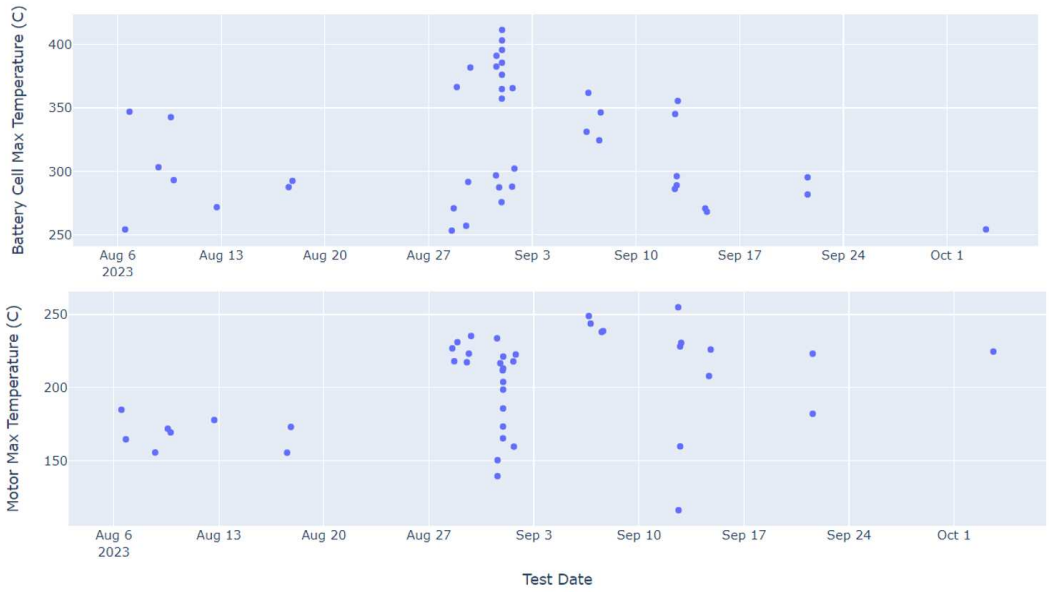

Fig. 4.7: Motor and Battery Temperature Trends During Takeoff

Landing Gear brake temperature is another critical signal that should be monitored during aircraft landings. similar to the exercise above, brake temperature data for all the landings over a two month period can be extracted using a simple SQL query. For this analysis, data is collected from four thermistors installed on the landing gear brakes, and the maximum temperature recorded from each thermistor is plotted as a data point. When plotting the temperature data (Figure 4.8) we do not observe any trends across the various flight tests. We also see that none of the thermistors is recording consistently hotter readings, which would be indicative of an issue.

Moreover, similar to the trend seen in the above plots, touch-and-go maneuvers do not allow enough time for the components to sufficiently cool down resulting in increasingly higher temperatures with each landing. This is clearly visible when focusing on the data from the test on August 31st. However, since brake application is minimal during this maneuver, the magnitude of temperature increase is small for each touch-and-go, with the

exception of the final landing of that flight test, where braking is used to reach a complete stop before taxiing off the runway.

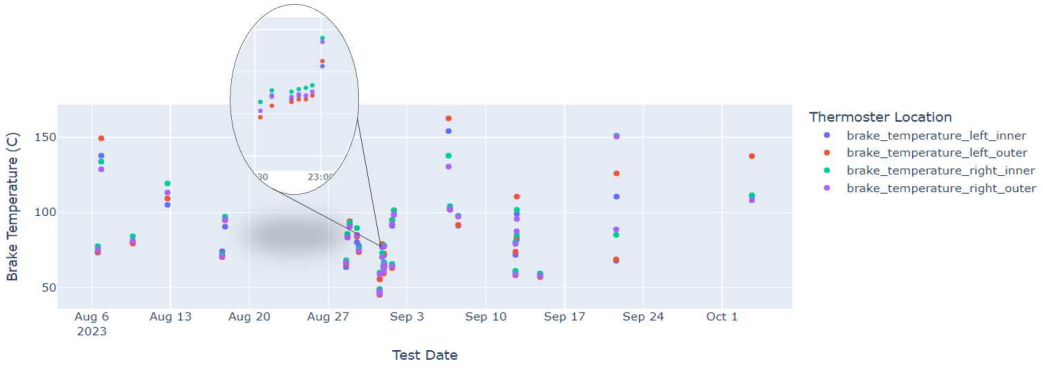

Fig. 4.8: Landing Gear Brake Temperatures

The final (and perhaps most obvious) trend monitoring use case for this phase classification model is aircraft efficiency during cruise. In conventional aircraft with an internal combustion engine, aircraft efficiency is typically measured using Specific Air Range (SAR). This metric indicates the amount of fuel burned per nautical mile and typically has units of lb/nm. For electric aircraft, we can derive a similar measure of efficiency by calculating the total power expended by the aircraft batteries per nautical mile flown. For this analysis, cruise phases identified by the phase of flight model are extracted along with the relevant aircraft signals. For each phase, the total power expended during the phase is divided by the total distance covered. The results can be seen below in Figure 4.9.

While most of the flights follow a similar trend, the flight test from September 1st is an obvious outlier. Further investigation into this particular flight test indicates that there were significant tailwinds during the flight. This is believed to be the reason behind the large offset in efficiency for that flight.

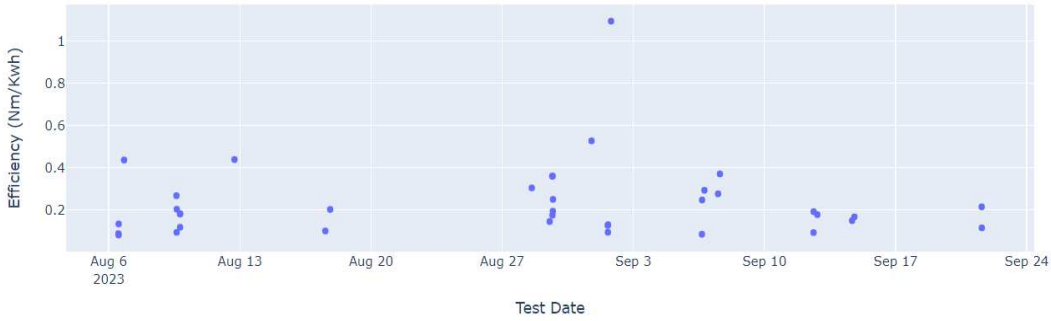

Fig. 4.9: Aircraft efficiency during cruise

The above static figures are useful for investigating trends and identifying outliers over the span of a range of time. However, even more added value comes from using the phase classification in automatic anomaly detection and report generation. After every flight test, plots similar to those above can be regenerated to determine if there are any changes in trends. Furthermore, values not within the expected range can be flagged for further analysis by the relevant engineers. While this workflow is developed for flight testing, it can also be useful for aircraft in operation where flight data is not as thoroughly analyzed. This automated report generation will be implemented as part of future work.

4.6 Model Presentation and Usability

Model classifications are only as useful as they are accessible. Therefore as part of model deployment and maintenance, considerable effort was put into making that the model classifications easily accessible by engineers and end users. All model classifications are stored in the same database where it is relatively easy to join on telemetry signals coming from the aircraft. This allows engineers to easily filter flight test data to answer complicated questions. For example, using the model predictions, engineers can ask questions such as:

'What is the aircraft response to abrupt pilot inputs during the different hover testing conducted?' The ability to quickly generate quantitative answers to such questions is critical for assessing pilot handling, motor performance, and other aircraft design aspects.

This accessibility is demonstrated below, where the user can see the different aircraft phases plotted against sample signals of interest collected from the aircraft alongside the aircraft geographical position. This same data can also be exported for further analysis in other programs (for ex: Matlab) as needed.

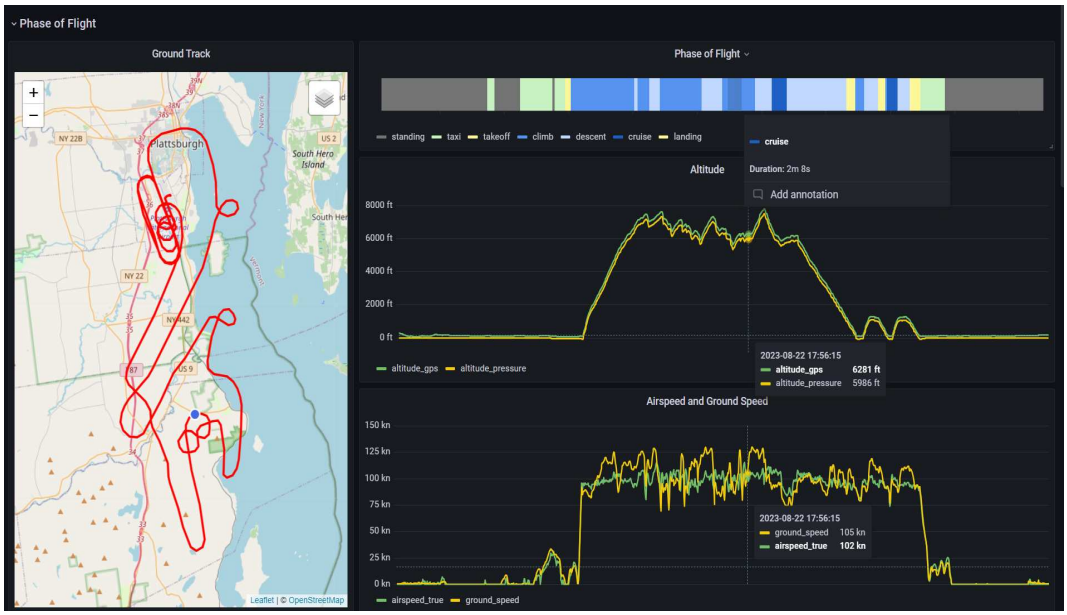

Fig. 4.10: Sample Visualization Panel

The above dashboard was created in Grafana, an open source dashboarding platform that can be used to query and visualize data from various data sources (44).

4.7 Flight Test Data Mining

The process described in Chapter 3.8 is a fairly simple way of ensuring that flight test data and its associated metadata is easily searchable and identifiable. This approach provides an automatically created list of flight tests that were conducted over a given time range. The engineer can then browse this list to find the relevant flight test event and download or view the data for their analysis.Similar to phase of flight visualization, a dashboard was built to allow analysts and engineers to easily access this data. An example snippet from that dashboard can be seen below.

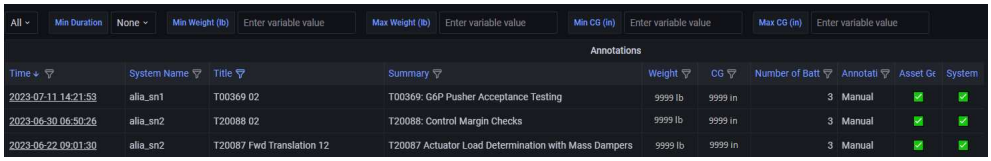

Fig. 4.11: Flight Event Finder Grafana Dashboard

As can be seen in Figure 4.12, in the top panel, a range of weight and CG can be input by the user to filter flight tests on metadata. In addition all the columns are searchable and can be filtered on. This allows users to filter on test summary, the number of battery packs, or any other column to find the necessary data. Clicking on a single row in that dashboard will lead to a more detailed page where the user can see all the associated metadata as well as other details pertaining to that flight test event. More importantly, assuming that the test summary is adequately filled by the FTE, this makes the flight test searchable via text queries. So an engineer can search for the word *stall* and all the flight test events where the word *stall* appeared in the test summary can be easily accessed. An example of this access pattern is shown below.

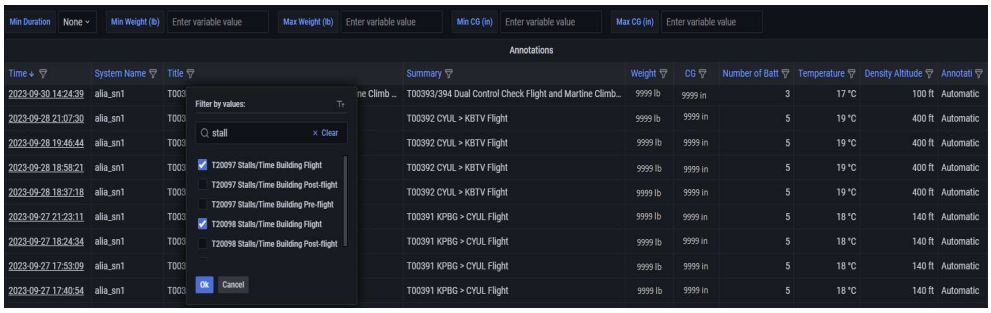

Fig. 4.12: Finding Flight Events with Stall Testing

In the last three months this automatic flight test detection process has identified 172 flight events where the majority was successfully associated with configuration data from Jira as well as flight test artifacts such as flight test reports and FTE notes. As is the case with phase classifications, this higher level information can be used to better understand the aircraft performance and conduct longitudinal analysis to identify trends. Figure 4.13 below shows the maximum aircraft speed, altitude, and rate of climbs achieved during each flight, plotted against weight data obtained from Jira. As can be seen in the plot below, aircraft weight is not a limiting factor on these performance parameter indicating that more flight envelope exploration is required to reach aircraft performance limits during flight test.

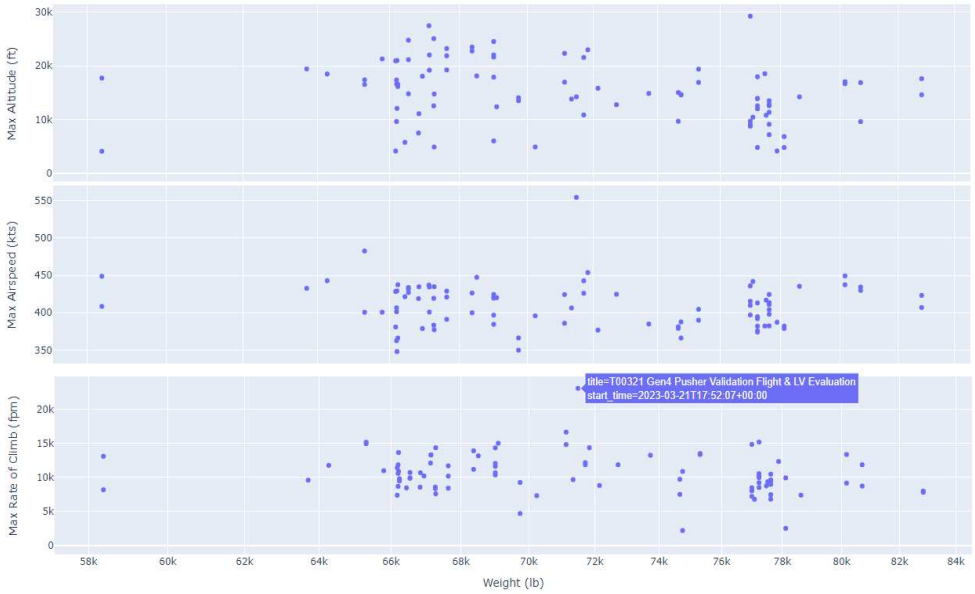

Fig. 4.13: Aircraft Weight vs. Max Altitude, Airspeed, and Rate of Climb

In order to aid with flight envelope exploration, we can plot the weight and CG data obtained from Jira against the aircraft Weight-CG envelope to visualize the parts of the envelope that still need to be explored via flight test. Figure 4.14 shows a representation of such a plot. As typically a flight test program starts in the middle of the envelope to minimize safety risks, this view can help plan upcoming tests to gradually expand the flight testing configuration towards the edges of the envelope.

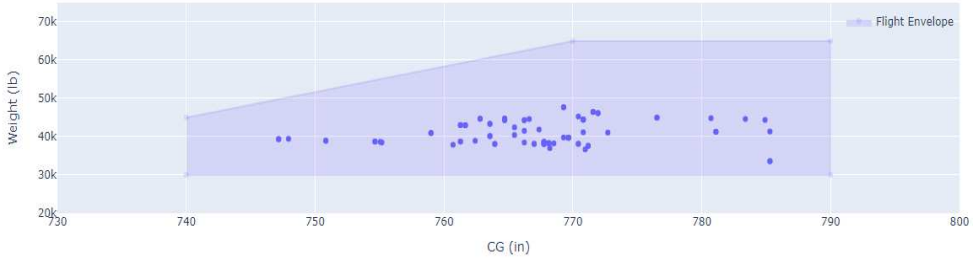

Fig. 4.14: Weight-CG Envelope

Finally we can combine phase of flight classifications, metadata obtained from Jira, and aircraft telemetry data to better understand the impact of weight and CG on aircraft

efficiency (SAR in conventional aircraft). Since both flight events and phase classifications are stored in the same database, we can merge aircraft telemetry data where the model determined the aircraft to be in cruise with CG and weight data obtained from Jira. Figure 4.15 shows the results of this aggregation. The details of calculating the efficiency metric is discussed in Section 4.5

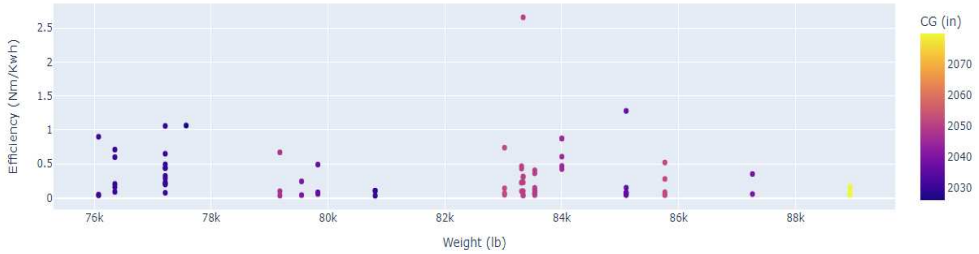

Fig. 4.15: Effect of Weight and CG on Aircraft Efficiency

Chapter 5

Conclusion

This research investigated methods for facilitating the analysis of flight test data and maximizing the value extracted from this vital data source during aircraft development and certification. In particular, the focus was on e-VTOL flight testing applications and how statistical modeling and data mining can be used to achieve the aforementioned objectives. An e-VTOL aircraft developed by Beta Technologies was the main data source for this research. Different statistical models were investigated for the task of classifying the different phases of flight during a flight test. In total, seven models were investigated and an exercise in hyperparameter tuning was conducted on the most promising models. Finally, the Gradient Boosting model with the quoted parameters was found to be the most performant given the performance metrics developed for this task. In addition, synthetic data generation was used to augment the training dataset for the LSTM model. This data was generated using a high-fidelity simulation model but the LSTM model performance was surprisingly degraded by the addition of this data. This indicates that perhaps higher quality simulation data that is more representative of real flight test data should be used to improve model

performance. Finally, rule based logic was developed to identify VTOL phases such as Hover, Transition-In, and Transition-Out. The small amount of data available and rigid phase definitions made a rule-based model a good candidate for this classification task and it was found to perform adequately on the available data. Model deployment strategies as well as model output accessibility was presented.

The second part of this research involved using the above model and a combination of different flight test data sources to provide contextual information around each data point from the aircraft. The first step is using the phase of flight model to determine when a flight has actually occurred (since data is recorded whenever the aircraft is on). Once flight start and end times are identified, the aircraft data is associated to metadata provided by the FTE such as test purpose, flight test reports, and other contextual information. Finally information retrieved from the flight test report data source is used to identify relevant aircraft and testing configurations from a tertiary data source. This source contains information such as aircraft CG, weight and conditions of the day (temperature, density altitude, winds, etc.).

The end result of the above processes is a robust workflow that allows engineers and analysts to navigate the terabytes of flight test data that the aircraft generates. It allows for easy filtering and searchability to ensure aircraft analysis and design decisions are backed by as much real-world data as possible. This also provides a singular destination that includes both the raw data collected from the aircraft, and contextual data provided by FTEs for analysis and visualization. As presented in the sample use cases, conducting longitudinal analysis and trend monitoring becomes relatively trivial. This allows for more thorough investigations and a better understanding of the available data collected during the flight test campaign.

5.1 Suggestions for Future Work

A surprising finding from this research was the lack of LSTM model performance improvement when synthetic data was added to the training dataset. In fact, the model showed degraded performance. This is an indication that the quality of the synthetic data should be improved. One way to do this is to further introduce stochasticity into the data to ensure that the generated missions are more representative of real flight test data. This can be achieved by setting up multiple mission profiles for which to vary the parameters described in Section 3.2. This way maneuvers such as Touch-and-go and go-arounds (which are commonly seen in flight test) can be incorporated into mission profiles. More importantly a controller capable of performing landings should be used to ensure that this critical phase is not neglected.

Another approach is to physically fly different mission profiles in the aircraft simulator to generate the mission profiles. While more expensive and time consuming, it will provide the most representative synthetic data. Using this approach, all the different maneuvers and flight conditions seen in flight test can be replicated as long as the simulator physics allows it. However, this data generation method requires an operator/pilot that is familiar with the aircraft controls and simulator operation. It also limits the amount of data that can be generated to the availability of the flight simulator which is typically also used for aircraft development and certification.

Another area of improvement from this research is the rule-based model used to identify VTOL phases. In particular, the model's reliance on the Radar Altitude signal. As discussed in Section 3.6.2, this signal is susceptible to data quality issues and thus negatively impacts the model classification performance. It is possible to eliminate this signal

completely and only use hover and pusher throttle signals as inputs to the algorithm. However, this is likely to result in misclassification during aircraft ground testing. It is also unlikely that an alternative altitude signal can be used, as the other signals typically don't report altitude as the height above ground, but rather as Pressure Altitude or height above Mean Sea Level (MSL). One option is to simply reduce the model's reliance on this signal relative to the two throttle signals. A simple weighted average can be used to that effect. In addition, some pre-processing can be done on the signal to alleviate the quality issues. A combination of filtering and interpolation can be used to deal with both data spikes and data drops in the Radar Altitude signal.

References

[1] F. Stoliker, *Introduction to Flight Test Engineering*. Flight Test Techniques Series, Research and Techonology Organization (RTO), North Atlantic Treaty Organization (NATO), 2005.

[2] B. Fox, M. Boito, J. C. Graser, and O. Younossi, *Test and Evaluation Trends and Costs for Aircraft and Guided Weapons*. RAND Corporation, November 2004.

[3] FutureFlight, "Beta Technologies eVTOL." `https://www.futureflight.aero/aircraft-program/beta-evtol?model=alia`. Accessed On: 2023-09-10.

[4] J. Fan and D. Li, "An Overview of Data Mining and Knowledge Discovery," *Journal of Computer Science and Technology*, vol. 13, pp. 348–368, July 1998.

[5] S. L. Brunton, J. Nathan Kutz, K. Manohar, A. Y. Aravkin, K. Morgansen, J. Klemisch, N. Goebel, J. Buttrick, J. Poskin, A. W. Blom-Schieber, T. Hogan, and D. McDonald, "Data-Driven Aerospace Engineering: Reframing the Industry with Machine Learning," *AIAA Journal*, vol. 59, pp. 2820–2847, Aug. 2021.

[6] D. O. Tipps, J. H. Rustenburg, and D. A. Skinn, "Statistical loads data for B-767-200ER Aircraft in Commercial Operations," Tech. Rep. URD-TR-99-00072, Univer-

sity of Daytona Research Institute and United States Federal Aviation Administration Office of Aviation Research, March 2000.

[7] V. Goblet, N. Fala, and K. Marais, "Identifying Phases of Flight in General Aviation Operations," in *15th AIAA Aviation Technology, Integration, and Operations Conference*, June 2015.

[8] D. J. Linse and R. F. Stengel, "Identification of aerodynamic coefficients using computational neural networks," *Journal of Guidance, Control, and Dynamics*, vol. 16, pp. 1018–1025, November 1993.

[9] Y. Li, H. Si, Y. Zong, X. Wu, P. Zhang, H. Jia, S. Xu, and D. Tang, "Application of Neural Network Based on Real-Time Recursive Learning and Kalman Filter in Flight Data Identification," *International Journal of Aeronautical and Space Sciences*, vol. 22, pp. 1383–1396, December 2021.

[10] A. Gavrilovski, H. Jimenez, D. N. Mavris, A. H. Rao, S. Shin, I. Hwang, and K. Marais, "Challenges and Opportunities in Flight Data Mining: A Review of the State of the Art," in *AIAA Infotech at Aerospace*, January 2016.

[11] V. Duggal, T.-N. Tran, D.-T. Pham, and S. Alam, "Modelling Aircraft Priority Assignment by Air Traffic Controllers During Taxiing Conflicts Using Machine Learning," in *2022 Winter Simulation Conference (WSC)*, December 2022.

[12] M. Salvador, S. Yacout, and A. AboElHassan, "Using Big Data and Machine Learning to Improve Aircraft Reliability and Safety," in *2022 Annual Reliability and Maintainability Symposium (RAMS)*, January 2022.

[13] K. Kirkpatrick, J. May, and J. Valasek, "Aircraft System Identification Using Artificial Neural Networks," in *51st AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition*, January 2013.

[14] R. Raman, B. R, P. Mahalingaiah, and M. Goswami, "Machine Learning Models for Aircraft Tail Performance & Degradation Predictions," in *2021 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT)*, July 2021.

[15] D. Wu, Z. Sun, Y. Zhu, L. Tian, H. Zhu, P. Xiong, Z. Cao, M. Wang, Y. Zheng, C. Xiong, H. Jiang, K. H. Tsoi, X. Niu, W. Mao, C. Feng, X. Zha, G. Deng, and W. Luk, "Custom machine learning architectures: towards realtime anomaly detection for flight testing," in *2018 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, IEEE, May 2018.

[16] E. Arts, "A Novel Approach to Flight Phase Identification using Machine Learning," Master's thesis, Hamburg University, 2021.

[17] S. Kovarik, L. Doherty, K. Korah, B. Mulligan, G. Rasool, Y. Mehta, P. Bhavsar, and M. Paglione, "Comparative Analysis of Machine Learning and Statistical Methods for Aircraft Phase of Flight Prediction," *International Conference on Research in Air Transportation 2020, 9th International Conference*, 2020.

[18] H.-J. Chin, A. Payan, C. Johnson, and D. N. Mavris, "Phases of Flight Identification for Rotorcraft Operations," in *AIAA Scitech 2019 Forum*, January 2019.

[19] R. Boushehri, R. Motamed, K. Ellison, and K. Stanton, "Estimating Epistemic Uncertainty in Soil Parameters for Nonlinear Site Response Analyses: Introducing the

Latin Hypercube Sampling technique," *Earthquake Spectra*, vol. 38, pp. 2422–2450, November 2022.

[20] A. A. Lambregts, "TECS Generalized Airplane Control System Design – An Update," in *Advances in Aerospace Guidance, Navigation and Control* (Q. Chu, B. Mulder, D. Choukroun, E.-J. Van Kampen, C. De Visser, and G. Looye, eds.), pp. 503–534, Berlin, Heidelberg: Springer Berlin Heidelberg, 2013.

[21] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[22] Wes McKinney, "Data Structures for Statistical Computing in Python," in *Proceedings of the 9th Python in Science Conference* (Stéfan van der Walt and Jarrod Millman, eds.), 2010.

[23] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "PyTorch: An Imperative Style, High-Performance Deep Learning Library," *Advances in Neural Information Processing Systems*, vol. 32, December 2019.

[24] NVIDIA and Vingelmann, Péter and Fitzek, "Cuda, release: 10.2.89." `https://developer.nvidia.com/cuda-toolkit`, 2020. Accessed On: 2023-05-19.

[25] M. Löning, A. Bagnall, S. Ganesh, and V. Kazakov, "Sktime: A Unified Interface for Machine Learning with Time Series," *arXiv preprint arXiv:1909.07872*, 2019.

[26] A. G. Bostrom, *Shapelet Transforms for Univariate and Multivariate Time Series Classification*. Doctoral Thesis, University of East Anglia, May 2018.

[27] C. M. Bishop, *Pattern Recognition and Machine Learning*. Information science and statistics, New York: Springer, 2006.

[28] L. Breiman, "Random Forests," *Machine Learning*, vol. 45, pp. 5–32, October 2001.

[29] J. H. Friedman, "Stochastic Gradient Boosting," *Computational Statistics & Data Analysis*, vol. 38, pp. 367–378, February 2002.

[30] M. Middlehurst, J. Large, M. Flynn, J. Lines, A. Bostrom, and A. Bagnall, "HIVE-COTE 2.0: A New Meta Ensemble for Time Series Classification," *Machine Learning*, vol. 110, pp. 3211–3243, December 2021.

[31] M. Middlehurst, J. Large, and A. Bagnall, "The Canonical Interval Forest (CIF) Classifier for Time Series Classification," in *2020 IEEE International Conference on Big Data (Big Data)*, December 2020.

[32] M. Middlehurst, J. Large, G. Cawley, and A. Bagnall, "The Temporal Dictionary Ensemble (TDE) Classifier for Time Series Classification," in *Machine Learning and Knowledge Discovery in Databases* (F. Hutter, K. Kersting, J. Lijffijt, and I. Valera, eds.), (Cham), pp. 660–676, Springer International Publishing, 2021.

[33] P. Schäfer, "The BOSS is Concerned with Time Series Classification in the Presence of Noise," *Data Mining and Knowledge Discovery*, vol. 29, November 2015.

[34] A. Dempster, F. Petitjean, and G. I. Webb, "ROCKET: Exceptionally Fast and Accurate Time Series Classification Using Random Convolutional Kernels," *Data Mining and Knowledge Discovery*, vol. 34, Sept. 2020.

[35] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A Next-Generation Hyperparameter Optimization Framework," in *Proceedings of the 25rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019.

[36] J. Bergstra, D. Yamins, and D. Cox, "Hyperopt: A Python Library for Optimizing the Hyperparameters of Machine Learning Algorithms," in *Python in Science Conference*, (Austin, Texas), 2013.

[37] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, pp. 182–197, April 2002.

[38] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, "Algorithms for Hyper-Parameter Optimization," in *Advances in Neural Information Processing Systems* (J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K. Q. Weinberger, eds.), vol. 24, Curran Associates, Inc., 2011.

[39] "What is a Container? | Docker." `https://www.docker.com/resources/what-container/`. Accessed On: 2023-09-04.

[40] C. Giebler, C. Gröger, E. Hoos, H. Schwarz, and B. Mitschang, "Leveraging the Data Lake: Current State and Challenges," in *Big Data Analytics and Knowledge Discovery* (C. Ordonez, I.-Y. Song, G. Anderst-Kotsis, A. M. Tjoa, and I. Khalil, eds.), pp. 179–188, Springer International Publishing, 2019.

[41] Atlassian, "Jira - Issue & Project Tracking Software." `https://www.atlassian.com/software/jira`. Accessed On: 2023-10-11.

[42] L. Espenschied and R. C. Newhouse, "A Terrain Clearance Indicator," *The Bell System Technical Journal*, vol. 18, pp. 222–234, January 1939.

[43] D. T. McRuer, "Pilot-Induced Oscillations and Human Dynamic Behavior," Tech. Rep. H-2042, July 1995. NTRS Author Affiliations: Systems Technology, Inc. NTRS Document ID: 19960020960 NTRS Research Center: Armstrong Flight Research Center (AFRC).

[44] G. Labs, "Grafana | Query, visualize, alerting observability platform." `https://grafana.com/grafana/`. Accessed On: 2023-09-04.